The Resource Journal for IBM Micro Programmers

PROGRAMMER'S Journal

MARCH/APRIL 1988
VOLUME 6.2
$5.00

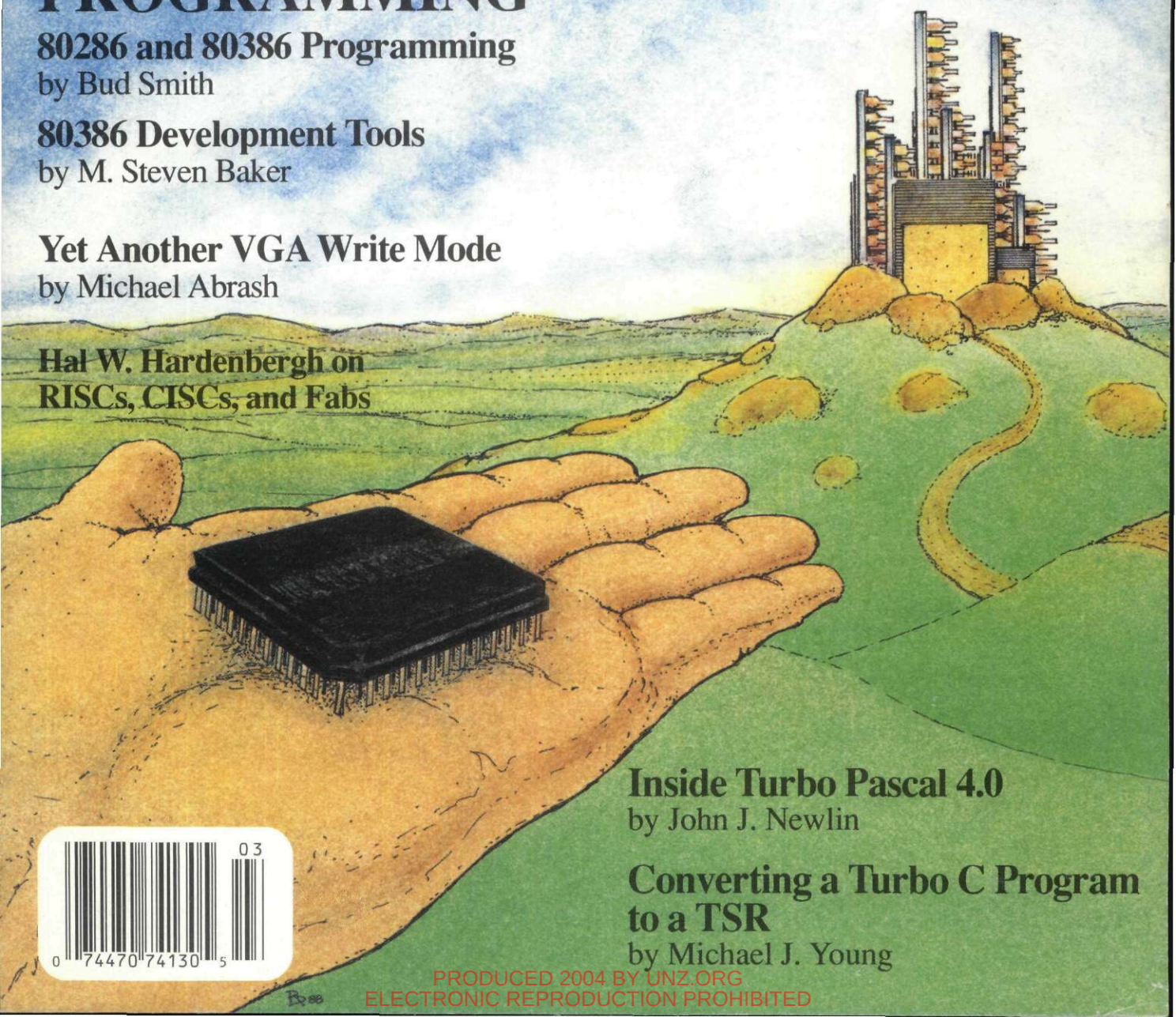
386 PROGRAMMING

80286 and 80386 Programming
by Bud Smith

80386 Development Tools
by M. Steven Baker

Yet Another VGA Write Mode
by Michael Abrash

Hal W. Hardenbergh on RISCs, CISCs, and Fabs



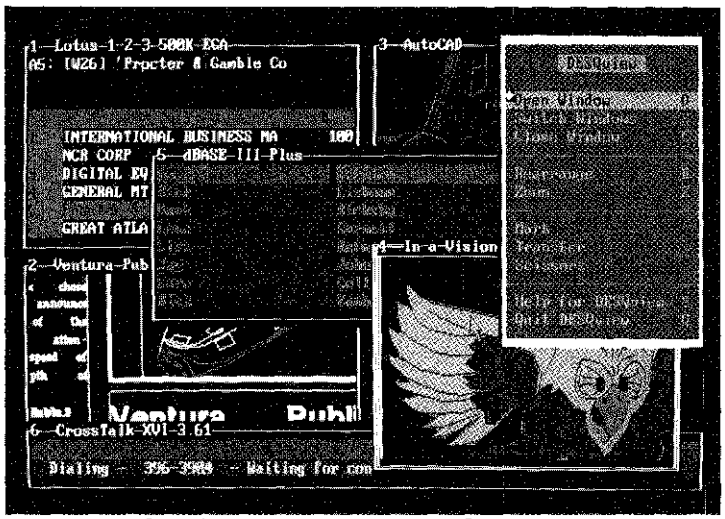
Inside Turbo Pascal 4.0
by John J. Newlin

Converting a Turbo C Program to a TSR
by Michael J. Young

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

How to tell the difference between DESQview™ 2.0 and any other environment.

Selecting DESQview, the environment of choice, can give you the productivity and power you crave, without the loss of your old programs and hardware. If you like your existing programs, want to use them together, transfer data between them, print, sort, communicate with or process-in-background, yet still have the need to keep in place your favorite PC(8088, 8086, 80286 or 80386), DESQview is the "proven true" multitasking, multi-windowing environment for you. Best of all, DESQview 2.0 is here now, with all the money saving, time saving, and productivity features that others can only promise for the all-too-distant future.



One picture is worth a thousand promises.

And with DESQview's new graphics enhancements for Hercules, CGA, EGA, and VGA, Version 2.0 still offers the same award winning and pioneering features for programs that earned DESQview its leadership, only now you can also run desktop publishing programs, CAD programs, even GEM-,™ Topview-,™ and Microsoft Windows-™ specific programs. In some cases you'll add as little as 10–40K to your system overhead. Now you can have multi-tasking, multi-windowing, break the 640K habit too and still get an auto dialer, macros, menus for DOS and, for advanced users, a new complete application programmer's interface capability. No wonder that over the years, and especially in recent months, DESQview, and now DESQview 2.0 have earned extravagant praise from some of the most respected magazines in the industry.

"Product of the Year" by readers vote in InfoWorld.

"Best PC Environment" by popular vote at Comdex Fall in PC Tech Journal's "System Builder" Contest.

"—I wouldn't want to run an IBM or compatible computer without DESQview"—InfoWorld, Michael Miller.

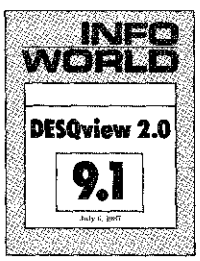
"A colossus among windowing environments"...."will run almost anything"—PC Week, Marvin Bryan.

"Windows, promises, but DESQview delivers"—MICRO-TIMES, Birrell Walsh.

No other environment has consistently pioneered features, openness, and productivity. See for yourself. Send in the coupon. The possibilities are endless with DESQview 2.0.

Attention Programmers: For more information about Quarterdeck's API, and future 386 program extensions, call us today.

SYSTEM REQUIREMENTS
IBM Personal Computer and 100% compatibles (with 8086, 8088, 80286 or 80386 processors) with monochrome or color display; IBM Personal System/2 • Memory: 640K recommended; for DESQview itself 0-145K • Expanded Memory (Optional): expanded memory boards compatible with the Intel AboveBoard; enhanced expanded memory boards compatible with the AST RAMpage • Disk: Two diskette drives or one diskette drive and a hard disk • Graphics Card (Optional): Hercules, IBM Color/Graphics (CGA), IBM Enhanced Graphics (EGA), IBM Personal System/2 Advanced Graphics (VGA) • Mouse (Optional): Mouse Systems, Microsoft and compatibles • Modem for Auto-Dialer (Optional): Hayes or Compatible • Operating System: PC-DOS 2.0-3.3; MS-DOS2.0-3.2 • Software: Most PC-DOS and MS-DOS application programs; programs specific to TopView 1.1, GEM 1.1 and Microsoft Windows 1.03 • Media: DESQview 2.0 is available on either 5¼" or 3½" floppy diskettes



Rush me DESQview 2.0! Today! PGMR-3/4

No. of Copies	Media 3½"/5¼"	Product	Retail Price ea.	Total
		DESQview 2.0	$129.95	$
		Shipping & Handling USA	$ 5.00	$
		Outside USA	$ 10.00	
		Sales Tax (CA residents)	6.5%	$
			Amount Enclosed	$

Payment: ☐ Visa ☐ MC ☐ AMEX ☐ Check

Credit Card: Valid Since ____/____ Expiration ____/____

Card Number:

Credit Card Name

Shipping Address

City State Zip Telephone

Mail to: Quarterdeck Office Systems, 150 Pico Boulevard, Santa Monica, CA 90405.
NOTE: If you own DESQview call us for a special upgrade offer, or send in your DESQview registration card. AST Special Edition users included.

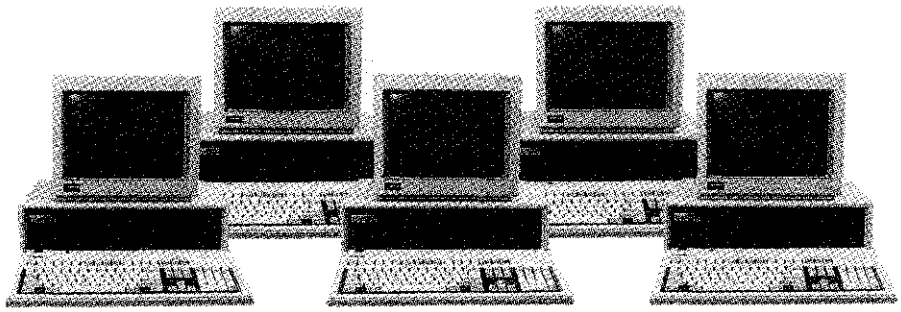
DESQ™ view
Quarterdeck

Quarterdeck Office Systems • 150 Pico Boulevard, Santa Monica, CA 90405 • (213) 392-9851

DESQview is a trademark of Quarterdeck Office Systems. AboveBoard is a trademark of Intel Corporation. Hayes is a trademark of Hayes MicroComputer Products Inc. IBM, PC, Personal System/2 and TopView are trademarks of International Business Machines Corporation. Microsoft Windows and MS are registered trademarks of Microsoft Corporation. Mouse Systems is a trademark of Metagraphics/Mouse Systems. RAMpage is a trademark of AST Research, Inc. GEM is a trademark of Digital Research. Hercules is a trademark of Hercules.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

VM/386



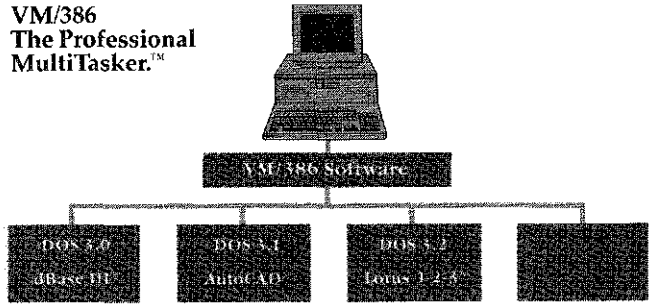
IT'S LIKE HAVING A DESKTOP FULL OF PCs

We know why you have it. Power. And now you can multiply that 386 power. Through true multitasking. With VM/386.™

VM/386 is the 80386 control program that brings you true DOS multitasking. VM/386 uses the virtual 8086 mode, built into the 80386 processor, to create individual virtual machines. You can load a different application in each virtual machine. It's like having a desktop *full of* PCs.

You have complete control over the virtual machines. You can tailor each virtual machine to fit your needs—and priorities. Each virtual machine has its own DOS, CONFIG.SYS, AUTOEXEC.BAT, and memory-resident programs along with its application. And each virtual machine is isolated from the others. A malfunction in one program doesn't destroy the others.

VM/386
The Professional
MultiTasker.™



There is virtually no limit to the number, or kind, of applications that can run simultaneously. For example, you can recalculate a 1-2-3® spreadsheet, sort a dBASE III® file, and receive your E-mail—all at the same time. You can even run more than one AutoCAD™ program simultaneously. And EGA applications run in the background as well as in the foreground—perfectly.

VM/386 protects your investment in software—and hardware. VM/386 will run all DOS software. Existing and future. Without modification. VM/386 is compatible with 80386-based PCs—either native or Intel® Inboard 386.™ VM/386 will support the monitors, hard disks, printers, mice, modems—even the network—you're currently using—or plan to use.

With VM/386 you use familiar commands. You don't have to buy new software or upgrades to get true multitasking. No PIF files. No special loaders. VM/386 works the way you do. Only lots faster.

VM/386 is easy to install. Easy to learn. And easy to use.

Dramatically increase your productivity. Start using all the power built into your 80386. Phone or write today.

IGC
4800 Great America Parkway
Santa Clara, California 95054
(408) 986-8373

System Requirements

80386-based computer such as COMPAQ® DESKPRO® 386® or 80286 computer with Intel® Inboard™ 386

One 1.2 Mb (5¼") disk drive or, one 3½" microfloppy

One hard disk drive

DOS 3.0 or later

2 Mb memory recommended

Not copy protected

Package includes both 5¼" and 3½" media.

VM/386 is a trademark of IGC.
AutoCAD is a trademark of Autodesk Inc.
COMPAQ and COMPAQ DESKPRO 386 are registered trademarks of Compaq Computer Corporation.
dBASE III is a registered trademark of Ashton-Tate Corporation.
Intel is a registered trademark of Intel Corporation.
Inboard is a trademark of Intel Corporation.
1-2-3 is a registered trademark of Lotus Development Corporation.

Number 169 on the Reader Service Card

IGC

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal for IBM Micro Programmers

PROGRAMMER'S Journal

Volume 6.2
March/April 1988

Contents

4 Notes from the Editors

6 Letters

14 PJ Forum: RISCs, CISCs, and Fabs *by Hal W. Hardenbergh*
Hal tilts at high-performance 32-bit chips and the production problem

20 Redirections: Pink Noise *by Frank D. Greco*
Frank mongers some rumors and tells us again and in more detail why he's so high on Sun

26 On Graphics: Yet Another VGA Write Mode *by Michael Abrash*
The easy stuff—write mode 2

38 Pascal: Inside Turbo Pascal 4.0 *by John J Newlin*
Once is not enough—John takes up after Tom Swan and recommends an overlay manager

47 C: The Proposed ANSI C Language Standard *by Rex Jaeschke*
Rex responds to a reader's questions

50 C: Converting a Turbo C Program to a TSR *by Michael J. Young*
Getting a Turbo C TSR ready for business—using Turbo C language extensions

64 386: 80286 and 80386 Programming *by Bud Smith*
Quick and dirty ports are not enough

68 386: 80386 Software Development *by M. Steven Baker*
More power for your programming—from Phar Lap, IGC, A.I. Architects, MetaWare, LPI, MicroWay, and SAIC

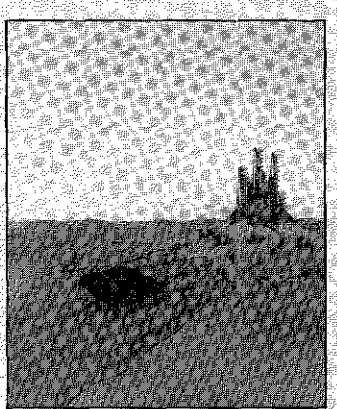
79 Review: The Norton Guide for Assembly Language *reviewed by Michael Abrash*
Help is at hand—off your bookshelf and onto your disk

80 Miscellaneous *by M. Steven Baker*
Program translators , communications protocols, the POSIX standard, GNU software, and the new *MS-DOS Encyclopedia*

86 PJ Billboard

86 Errata

88 Advertiser Index



RISCs, CISCs, the 286, the 386, the VGA— they're all here in this issue, and artist Ralph Bentley even included the Motorola on aesthetic grounds. That's an appropriate choice, too, as we welcome Hal Hardenbergh back to ***PJ***. ♦

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Create Magic with Blaise.

Turbo C TOOLS™

Magic is easy with Turbo C TOOLS in your bag of tricks. New Turbo C TOOLS™ from Blaise Computing is a library of compiled C functions that allows you full control over the computer, the video environment, and the file system, and gives you the jump on building programs with Borland's new C compiler. Now you can concentrate on the creative parts of your programs.

The library comes with well-documented source code so that you can study, emulate, or adapt it to your specific needs. Blaise Computing's attention to detail, like the use of function prototyping, cleanly organized header files, and a comprehensive, fully-indexed manual, makes Turbo C TOOLS the choice for experienced software developers as well as newcomers to C.

Turbo C TOOLS provides the sophisticated, bullet-proof capabilities needed in today's programming environment, including removable windows, "sidekickable" applications, and general interrupt service routines written in C.

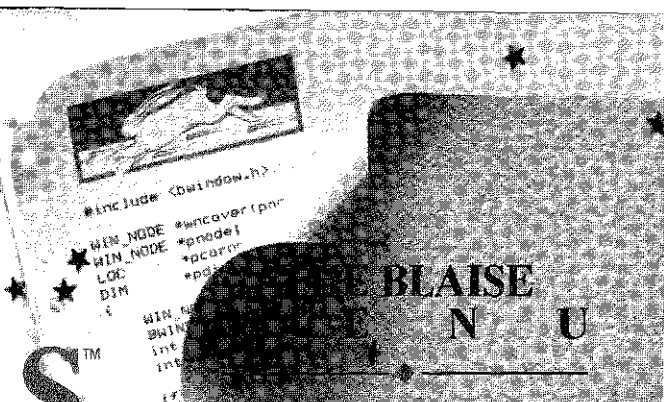
The functions contained in Turbo C TOOLS are carefully crafted to supplement Turbo C, exploiting its strengths without duplicating its library functions. As a result you'll get functions written predominantly in C, that isolate hardware independence, and are small and easy to use.

Turbo C TOOLS embodies the full spectrum of general purpose utility functions that are critical to today's applications. Some of the features in Turbo C TOOLS are:

- **WINDOWS** that are stackable and removable, that have optional borders and a cursor memory, and that can accept user input.
- **INTERRUPT SERVICE ROUTINE** support for truly flexible, robust and polite applications. We show you how to capture DOS critical errors and keystrokes.
- **INTERVENTION CODE** lets you develop memory resident applications that can take full advantage of DOS capabilities. With simple function calls, you can schedule a Turbo C function to execute either when a "hot key" is pressed or at a specified time.
- **RESIDENT SOFTWARE SUPPORT** lets you create, detect, and remove resident utilities that you write with Turbo C TOOLS.
- **FAST DIRECT VIDEO ACCESS** for efficiency, and support for all monitors including EGA 43-line mode.
- **DIRECTORY AND FILE HANDLING** support let you take advantage of the DOS file structure, including volume labels and directory structure.

Blaise Computing Inc. has a full line of support products for Turbo C, Microsoft C, Turbo Pascal and Microsoft Pascal. Call today for details, and make magic!

Turbo C TOOLS supports all versions of Turbo C including 1.5, requires DOS 2.00 or later and is just $129.00.



BLAISE
N U

Turbo POWER SCREEN *$129.00*
NEW! General screen management: paint screens; block mode data entry or field-by-field control with instant screen access. Now for Turbo Pascal 4.0, soon for C and BASIC.

Turbo POWER TOOLS PLUS *$129.00*
NEW VERSION! Now supports Turbo Pascal 4.0. Screen, window, and menu management including EGA support; DOS memory control; ISRs; scheduled intervention code; and much more.

Turbo ASYNCH PLUS *$129.00*
NEW VERSION! Now supports Turbo Pascal 4.0. Interrupt driven support for the COM ports; I/O buffers up to 64K; XON/XOFF; up to 19.2K baud; modem control and XMODEM file transfer.

C TOOLS PLUS *$129.00*
Full spectrum of general service utility functions including: windows; menus; memory resident applications; interrupt service routines; intervention code; and direct video access for fast screen handling. Specifically designed for Microsoft C 5.0 and QuickC.

ASYNCH MANAGER *$175.00*
Full featured interrupt driven support for the COM ports. I/O buffers up to 64K; XON/XOFF; up to 9600 baud; modem control and XMODEM file transfer. For Microsoft C and Turbo C or MS Pascal.

PASCAL TOOLS/TOOLS 2 *$175.00*
Expanded string and screen handling; graphics routines; memory management; general program control; DOS file support and more. For MS-Pascal.

Key Plus *$49.95*
"Super-batch" program. Create batch files which can invoke programs and provide input to them; run any program unattended; create demonstration programs; analyze keyboard usage.

EXEC *$95.00*
NEW VERSION! Program chaining executive. Chain one program from another in different languages; specify common data areas; less than 2K of overhead.

RUNOFF *$49.95*
Text formatter for all programmers. Written in Turbo Pascal; flexible printer control; user-defined variables; index generation; and a general macro facility.

TO ORDER CALL TOLL FREE
800-333-3087
TELEX NUMBER 338139

YES! I want to make magic!
Enclosed is $_____ for _____ copies of _____.
☐ Please send me more information on your products.
CA residents add Sales Tax. Domestic orders add $4.00 for UPS shipping, $10.00 for Federal Express standard air.
Name: _____ Phone: (_____) _____
Address: _____ State: _____ Zip: _____
City: _____ Exp. Date: _____
VISA or MC#: _____

MS-DOS and QuickC are registered trademarks of Microsoft Corporation. Turbo C and Turbo Pascal are registered trademarks of Borland International.

BLAISE COMPUTING INC.

2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

Number 2 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal for IBM PC/MS-DOS Programmers
PROGRAMMER'S Journal

Editors
Charles Aylworth, Erin O'Connor
Technical Editor
M. Steven Baker
Contributing Editors
Frank D. Greco, Tom Swan

Contributing Writers
Michael Abrash
M. Steven Baker
Frank D. Greco
Hal W. Hardenbergh
Rex Jaeschke
John J. Newlin
Bud E. Smith
Michael J. Young

Publisher
Elizabeth Oakley
Production
Ralph Bentley
Illustration
Jim Carpenter
Advertising
Robert Tappan
Subscriptions
Gary Hoyer
Accounting
Bobbi Sinyard
Lori Fanslow

Founder
Greg Estes

Business and Editorial Offices
Programmer's Journal
150 North 4th Street
Springfield, OR 97477
(503) 747-0800

Subscriptions: 1 year (6 issues) $24
($30 in Canada and $40 elsewhere)
Change of Address
Programmer's Journal
P.O. Box 30160
Eugene, OR 97403

Articles and letters in Programmer's Journal represent the views of their authors and not necessarily those of the publisher. Letters to the editor become the property of Programmer's Journal and may be edited and published.

Code and listings in Programmer's Journal appear "as is," and Programmer's Journal makes no guarantee or warranty of their quality, performance, accuracy, or suitability for any particular purpose. Programmer's Journal assumes no responsibility or liability to any person or entity with respect to any damage caused or alleged to be caused by their use.

Advertising: For ad rates, deadlines, and placement, call Robert Tappan (503/747-0800), or write to Programmer's Journal, P.O. Box 30160, Eugene, OR 97403.

Programmer's Journal is produced on Xerox Ventura Publisher, courtesy of Xerox Corporation

Programmer's Journal (ISSN 0747-586) is published bi-monthly by Oakley Publishing Company. Second-class postage paid at Eugene, OR, and additional mailing offices.

POSTMASTER: Send address changes to Programmer's Journal, P.O. Box 30160, Eugene, OR 97403.

© 1988 OAKLEY PUBLISHING COMPANY

Note from the Editors

I'm sorry to say, this is the last time I'll be addressing you as an editor of *Programmer's Journal*. The last eight months have been fun. I feel that I have been really privileged to have had the chance to talk with many *PJ* readers and work with the fine collection of authors who write for the magazine.

It's an exciting time to be working with the IBM microcomputers. As time goes on, and OS/2 matures, it should continue to be very interesting (in the Chinese sense). I've talked to a lot of developers, hardware and software, in the last few months. The array of products and ideas to be offered to the public in the coming year is truly astounding.

Erin O'Connor will be continuing as *PJ*'s editor. She is dedicated to bringing you the best and most useful material possible. There is always more that we wish we could publish, but within the limits of time, space, and money you'll be getting the best value possible for your subscription dollar.

So, Hail and farewell. I'll miss you all. It has been fun.—*Chuck*

Close to the end of his tenure as Editor of *PJ*, Robert Keller and I met at the behest of our publisher and an assortment of PR firms and directory compilers to pound out an editorial description of *PJ*. Robert had always been reluctant to pin us down to a stated, specific orientation. He didn't want to limit the range of articles submitted to *PJ* or paint us into a corner.

In May 1987, Robert was understandably more reluctant than ever. OS/2 had just been announced, the PS/2 and the VGA had just been unveiled, and the full powers of the 386 languished for want of an operating system. In the face of all this uncertainty, we managed to put together a cautious, interim, editorial statement—brief and peppered with the hedging word "currently."

The interim's lasted longer than we expected, and while some conclusions were foregone—Michael Abrash never missed a beat, plunging right in with a series on the VGA as packed with instruction as his classic set of EGA articles, and Chris Walker was pretty sure from the outset where he stood with respect to OS/2 and got busy on the 386—we're still not sure what the upshot will be.

Last November we asked in our Reader Survey whether you were developing for OS/2. The other day, Robin Tappan and I did a quick count of your answers to that question. Your response was an overwhelming "No" (85%)—or was that a surprisingly hefty "Yes" (15%, after all)? We think we'd better ask you again, now that you've had a few months to think it over and watch other developments. The same question appears on your Peer Review card for this issue.

One benefit of the uproar of the last few months is a new fluidity among categories of computers, and Steve Baker was ready to dive in. In our last issue, Steve had us porting large mainframe programs in FORTRAN to the increasingly powerful PC, and in this issue he looks at some interregnum products from minicomputer vendors that allow DOS to address the 386 in native mode. As usual, Steve's article is chock full, worth reading for the incidental information alone. In a companion piece on real mode programming for the 80386, Bud Smith tells us why it should be more than just a matter of quick ports from the 8086.

Frank Greco has always encouraged the breakdown of categories, and he seems positively gleeful as his areas of interest begin to encroach on the DOS world. Frank's at the same old stand, covering Apple, NeXT, Sun, insisting that workstations are PCs, and more than ever sure that good old Unix will have its day.

*

Paul Barkley seems to strike a responsive chord whether he speaks from his consulting office or the workplace. Paul's column for this issue arrived too late (Paul's juggling 45 hours per week for his employer with his consulting practice and his regular column for us). We're holding his column for next issue. It's a good one—"Getting Your Consulting Tools in Order."

*

Although it's a paradise in the summer, Eugene's off the beaten path and not too many people show up for the winter rains. To keep in touch, we have to leave the provinces once in awhile. We're flying south to San Francisco for the PC Faire April 7–10. Hope to see you there.

*

Robert Keller, by the way, engineered one conclusive outcome over the last nine months—Aaron Charles was born January 13th.—*Erin* ♦

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DEBUGGING SWAT TEAM

Order Eco-C88 Rel. 4.0 New Modeling Compiler and get C-more at no extra charge!

Seek and Correct

You already know that fast compilation does not mean fast program development. Backing up for bogus error messages and removing the bugs takes time. Eco-C88's "Seek and Correct" three - way error checking finds even the most elusive bugs, clearing the path for swift program development.

Double Barrel Error Checking

Eco-C88 nails syntax errors cold and tells you about the error in plain English. And there's no avalanche of false error messages, either. Other compilers can generate up to four times the number of error messages actually present; they leave it up to you to guess which ones are real. You'll be more productive with Eco-C88 because there is no guess work.

Eco-C88 provides ten levels of semantic error checking. You can select from almost no checking to the fussiest you've ever seen. Eco-C88's "picky flag" finds subtle errors that slip by other compilers.

Eco-C88 also features:

- All data types, plus ANSI Enhancements
- Robust library, including many new ANSI functions
- CED editor with online function help, split windows, compile-edit-link capability
- New, expanded manual with sample programs for the library functions

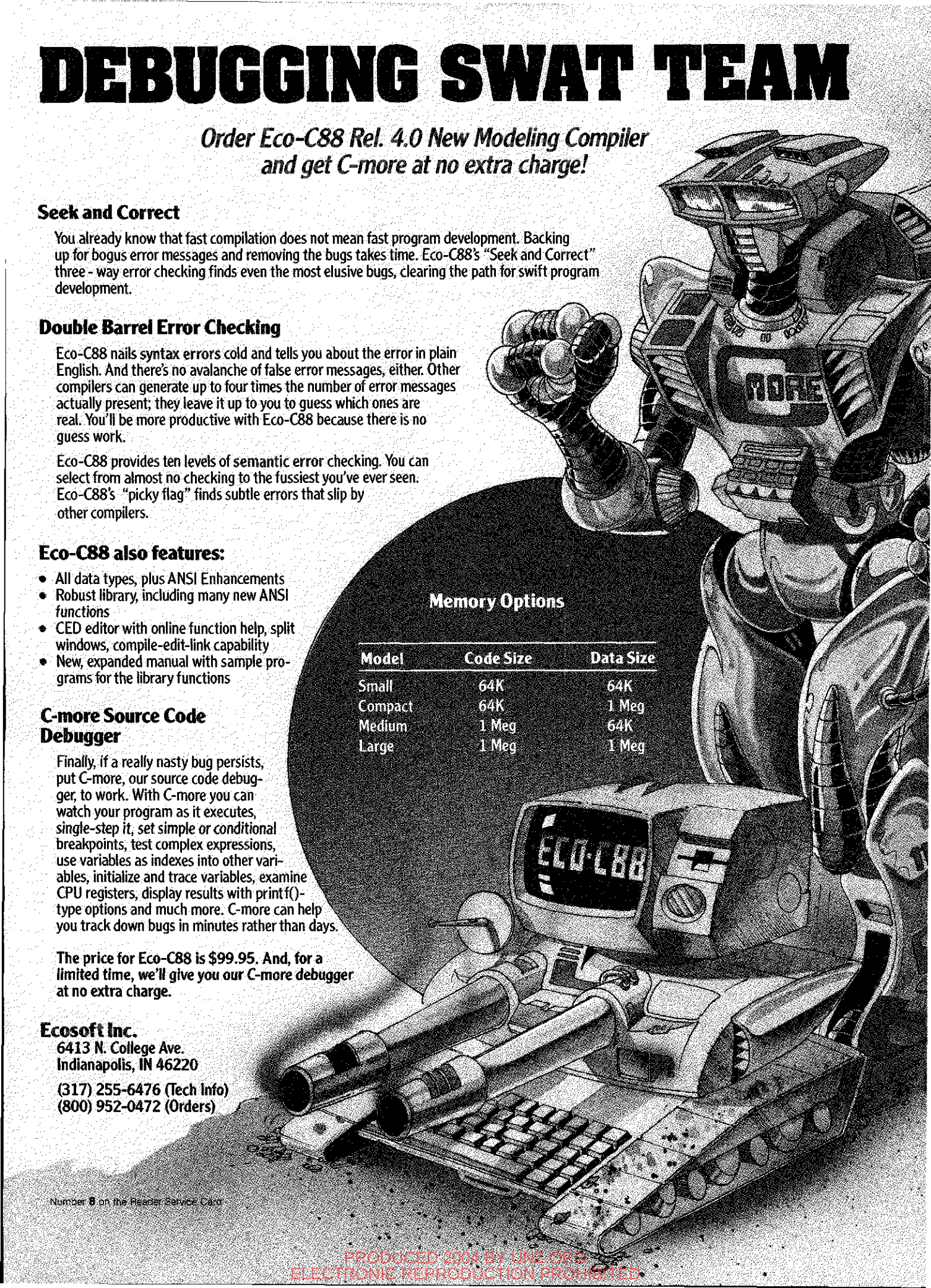
C-more Source Code Debugger

Finally, if a really nasty bug persists, put C-more, our source code debugger, to work. With C-more you can watch your program as it executes, single-step it, set simple or conditional breakpoints, test complex expressions, use variables as indexes into other variables, initialize and trace variables, examine CPU registers, display results with printf()-type options and much more. C-more can help you track down bugs in minutes rather than days.

The price for Eco-C88 is $99.95. And, for a limited time, we'll give you our C-more debugger at no extra charge.

Memory Options

Model	Code Size	Data Size
Small	64K	64K
Compact	64K	1 Meg
Medium	1 Meg	64K
Large	1 Meg	1 Meg



Ecosoft Inc.
6413 N. College Ave.
Indianapolis, IN 46220

(317) 255-6476 (Tech Info)
(800) 952-0472 (Orders)

Number 8 on the Reader Service Card.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

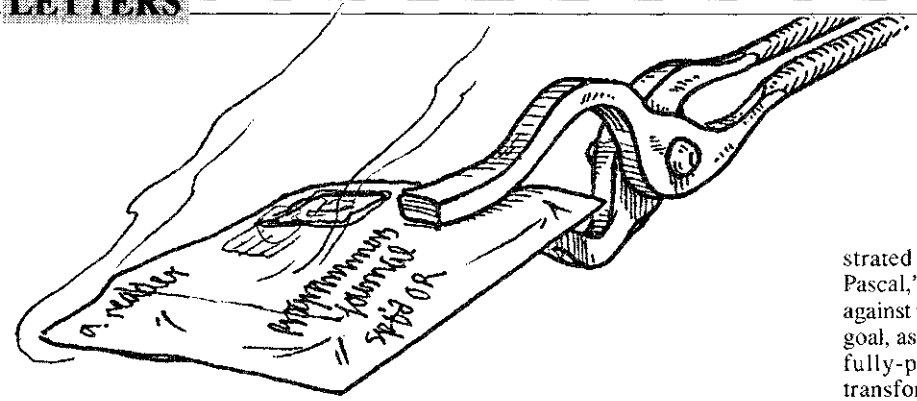


Diagram for the EGA

After reading Michael Abrash's articles on the EGA (*PJ*, Volumes 5.1–5.4) and carefully rereading the *IBM Tech Ref* for the EGA Adapter, I came up with the enclosed diagram. It shows functionally what happens when you read or write to the EGA display in graphics mode for plane 0—the other planes are identical. This helps me immensely when I'm programming at the hardware level.

Please review it. You are free to publish it if you think it would help your readers.

Bob Montgomery
Orlando, FL

Michael Abrash Replies

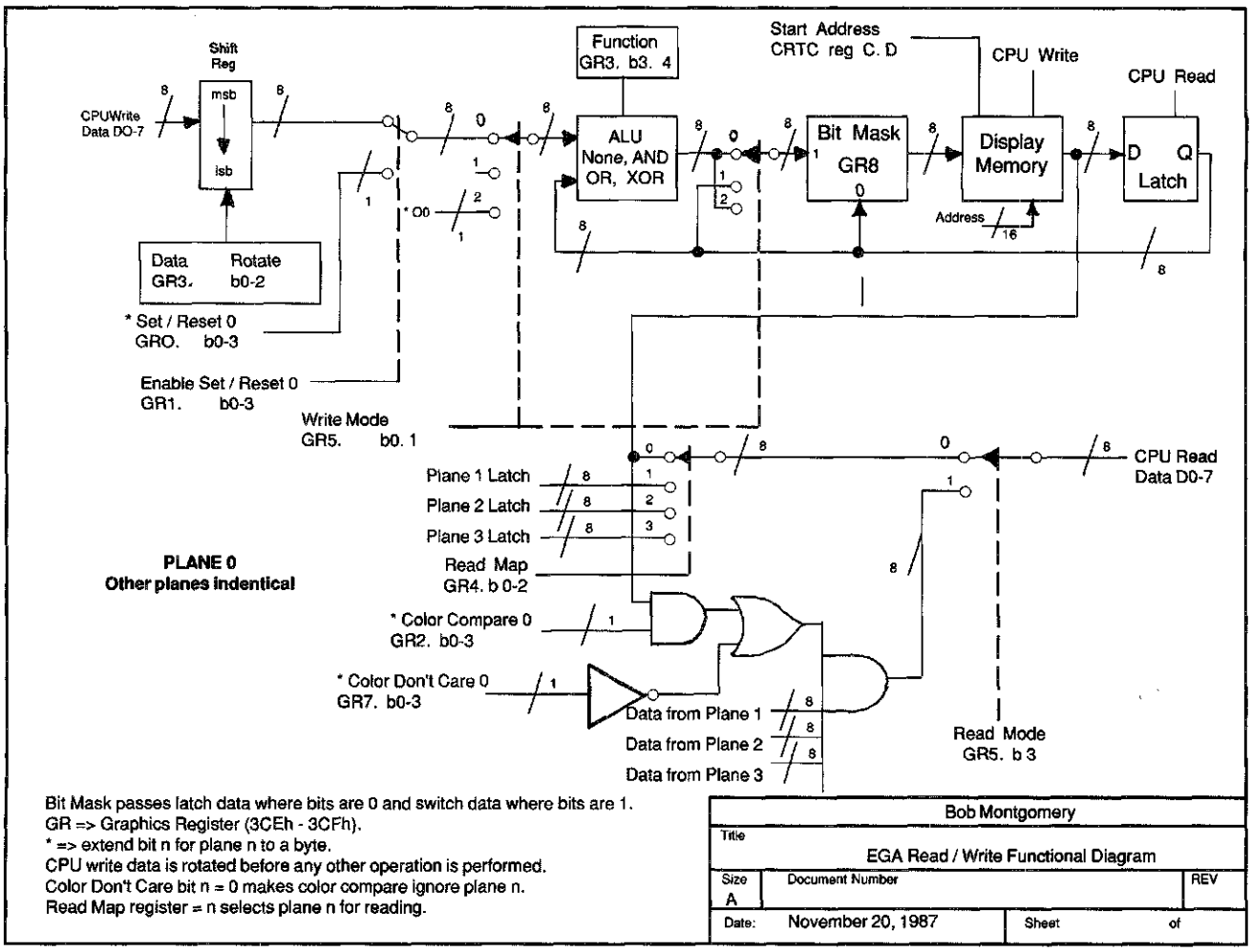
I'm no hardware engineer, but Bob Montgomery's diagram looks great to me. It should help a lot of people understand the EGA.

Michael Abrash
Mountain View, CA

Jon Greenblatt Goes Too Far (Even for Jon Greenblatt)

I'd like to respond to some comments in the Letters column in Volume 5.6. I would argue that "ludicrous" and "appalling" optimizations (readers' descriptions of those I demonstrated in "A Note on Optimizing Turbo Pascal," Volume 5.5) have to be measured against what you're trying to achieve. If your goal, as was ours at ELFSOFT, is to write a fully-programmable, self-customizing, transformational natural language parser, with a back-end that generates efficient dBASE code, there are some serious constraints. To be co-resident with dBASE (on 512K systems) required limiting ourselves to 250K. Furthermore, according to Winograd, Schank, Sager, *et. al.*, this program is impossible on its face—a futuristic dream. We weren't content merely to demonstrate it theoretically, but wanted to make it practical, which meant turbo-charging it until a complex query that generates two screens of .PRG code would translate in (average) 15 seconds.

This was necessary because however ludicrous or appalling it is, a manager who asks an employee to find out "how many of the *Programmer's Journal* prospects come


Shift Reg
msb
lsb
CPUWrite Data D0-7
Data GR3. Rotate b0-2
* Set / Reset 0 GR0. b0-3
Enable Set / Reset 0 GR1. b0-3
Write Mode GR5. b0. 1
Function GR3. b3. 4
ALU None, AND OR, XOR
Start Address CRTC reg C. D
Bit Mask GR8
CPU Write
CPU Read
Display Memory
Address
D Q Latch
Plane 1 Latch
Plane 2 Latch
Plane 3 Latch
Read Map GR4. b 0-2
PLANE 0
Other planes indentical
* Color Compare 0 GR2. b0-3
* Color Don't Care 0 GR7. b0-3
Data from Plane 1
Data from Plane 2
Data from Plane 3
CPU Read Data D0-7
Read Mode GR5. b 3
Bit Mask passes latch data where bits are 0 and switch data where bits are 1.
GR => Graphics Register (3CEh - 3CFh).
* => extend bit n for plane n to a byte.
CPU write data is rotated before any other operation is performed.
Color Don't Care bit n = 0 makes color compare ignore plane n.
Read Map register = n selects plane n for reading.
Bob Montgomery
Title
EGA Read / Write Functional Diagram
Size A
Document Number
REV
Date: November 20, 1987
Sheet
of

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Upgrade Your Technology

We're Programmer's Connection, the leading independent dealer of quality programmer's development tools for IBM personal computers and compatibles. We can help you upgrade your programming technology with some of the best software tools available.

Comprehensive Buyer's Guide. The CONNECTION, our new Buyers Guide, contains prices and up-to-date descriptions of *over 600 programmer's development tools by over 200 manufacturers*. Each description covers major product features as well as special requirements, version numbers, diskette sizes, and guarantees.

How to Get Your FREE Copy: 1) Use the reader service card provided by this journal; 2) Mail us a card or letter with your name and address; or 3) Call one of our convenient toll free telephone numbers.

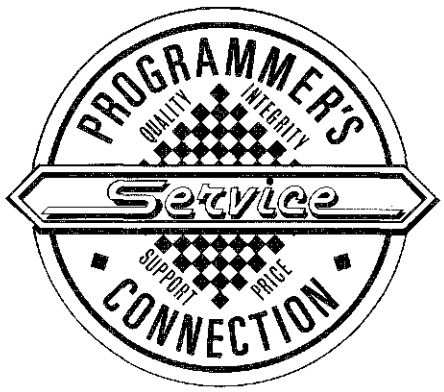
If you haven't yet received your copy of the Programmer's Connection Buyer's Guide, act now. Upgrading your programming technology could be one of the wisest and most profitable decisions you'll ever make.

USA..... 800-336-1166

Canada 800-225-1166
Ohio & Alaska (Collect) 216-494-3781
International.....216-494-3781
TELEX..... 9102406879

Business Hours: 8:30 AM to 8:00 PM EST Monday through Friday
Prices, Terms and Conditions are subject to change.
Copyright 1988 Programmer's Connection Incorporated

Sale Prices effective through 03/31/88



ORDERING INFORMATION

FREE SHIPPING. Orders within the USA (including Alaska & Hawaii) are shipped FREE via UPS. Call for express shipping rates.

NO CREDIT CARD CHARGE. VISA, MasterCard and Discover Card are accepted at no extra cost. Your card is charged when your order is shipped. Mail orders please include expiration date and authorized signature.

NO COD OR PO FEE. CODs and Purchase Orders are accepted at no extra cost. No personal checks are accepted on COD orders. POs with net 30-day terms (with initial minimum order of $100) are available to qualified US accounts only.

NO SALES TAX. Orders outside of Ohio are not charged sales tax. Ohio customers please add 5% Ohio tax or provide proof of tax-exemption.

30-DAY GUARANTEE. Most of our products come with a 30-day documentation evaluation period or a 30-day return guarantee. Please note that some manufacturers restrict us from offering guarantees on their products. Call for more information.

SOUND ADVICE. Our knowledgeable technical staff can answer technical questions, assist in comparing products and send you detailed product information tailored to your needs.

INTERNATIONAL ORDERS. Shipping charges for International and Canadian orders are based on the shipping carrier's standard rate. Since rates vary between carriers, please call or write for the exact cost. International orders (except Canada), please include an additional $10 for export preparation. All payments must be made with US funds drawn on a US bank. Please include your telephone number when ordering by mail. Due to government regulations, we cannot ship to all countries.

MAIL ORDERS. Please include your telephone number on all mail orders. Be sure to specify computer, operating system, diskette size, and any applicable compiler or hardware interface(s). Send mail orders to:

Programmer's Connection
Order Processing Department
7249 Whipple Ave NW
North Canton, OH 44720

		List	Ours
386 products			
386 ASM/386 LINK	*by Phar Lap*	495	**389**
386 DEBUGGER	*by Phar Lap*	195	**129**
PC/MOS	*All Varieties*	*New*	CALL	**CALL**
NDP C-386	*by MicroWay*	*New*	595	**529**
NDP ForTran-386	*by MicroWay*	*New*	595	**529**
blaise products			
ASYNCH MANAGER	*Specify C or Pascal*		175	**135**
C TOOLS PLUS/5.0			129	**99**
LIGHT TOOLS	*for Datalight C*	*Sale*	100	**55**
PASCAL TOOLS/TOOLS 2			175	**135**
RUNOFF	*Text Formatter*		50	**45**
Turbo ASYNCH PLUS/4.0			129	**99**
Turbo C TOOLS			129	**99**
Turbo POWER TOOLS PLUS/4.0			129	**99**
VIEW MANAGER	*Specify C or Pascal*		275	**199**
borland products			
EUREKA	*Equation Solver*		167	**105**
Quattro: The Professional Spreadsheet		*New*	195	**125**
Reflex: The Analyst			150	**99**
Sidekick			85	**57**
Superkey			100	**64**
Turbo Basic Compiler			100	**64**
Turbo Basic Database Toolbox			100	**64**
Turbo Basic Editor Toolbox			100	**64**
Turbo Basic Telecom Toolbox			100	**64**
Turbo C Compiler	*(Call for support products)*		100	**64**
Turbo Lightning			100	**64**
Turbo Lightning Word Wizard			70	**47**
Turbo Pascal		*Sale*	100	**59**
Turbo Pascal Database Toolbox			100	**64**
Turbo Pascal Developer's Toolkit			395	**259**
Turbo Pascal Editor Toolbox			100	**64**
Turbo Pascal Gameworks Toolbox			100	**64**
Turbo Pascal Graphix Toolbox			100	**64**
Turbo Pascal Numerical Methods Toolbox			100	**64**
Turbo Pascal Tutor			70	**41**
Turbo Prolog Compiler			100	**64**
Turbo Prolog Toolbox			100	**64**
c language			
C Programmer's Toolbox Combo	*by MMC AD*	*New*	130	**115**
C Programmer's Toolbox	*Specify Vol I or II*	*New*	80	**69**
C-terp	*by Gimpel, Specify compiler*	*Sale*	298	**199**
	for XENIX	*Sale*	498	**299**
dBx	*dBASE to C Translator by Desktop AI*		350	**299**
	with Source Code		550	**419**
DC88	*by C Ware*	*New*	99	**89**
DC88	*with Large Case Option by C Ware*	*New*	178	**159**
DMS Resident-C	*by American Software Intl*		150	**139**
Eco-C88 Modeling Compiler	*by Ecosoft*	*Sale*	100	**69**
Lattice C Compiler	*vers. 3.2 from Lattice*		500	**265**
PC Lint	*by Gimpel Software*		139	**99**
Turbo C-terp	*for Turbo C by Gimpel*	*New,Sale*	139	**99**
Vitamin C	*by Creative Programming*		225	**149**
VC Screen	*Forms Designer*		100	**79**
WKS LIBRARY	*by Tenon Software*		89	**79**
database management			
Clipper	*by Nantucket*		695	**379**
dBASE III Plus	*by Ashton-Tate*		695	**389**
Fox Base Plus	*by Fox Software*		395	**249**
Genifer	*by Bytel*		395	**249**
Paradox 1.1	*by Ansa/Borland*		495	**359**
Paradox 2.0	*by Ansa/Borland*		725	**525**
Paradox Network Pack	*by Ansa/Borland*		995	**725**
R:Base 5000	*by Microrim*		495	**359**
R:Base System V	*by Microrim*		700	**439**
Tom Rettig's Library	*by Tom Rettig & Assoc*		100	**79**
laboratory microsystems products			
CFORTH			200	**149**
FORTH/83 Metacompiler	*Specify Target*		750	**599**
PC/FORTH			150	**109**
PC/FORTH+			250	**188**
Programmer's Package #1			250	**199**
Programmer's Package #2			350	**279**
Programmer's Package #3			500	**399**
UR/FORTH	*Also Available for OS/2*		350	**258**
UR/FORTH Libraries			500	**395**
metagraphics products			
MetaWINDOW	*No Royalties*		195	**159**
MetaWINDOW/PLUS			275	**229**
QuickWINDOW/C	*for Microsoft QuickC*		95	**79**
TurboWINDOW/C	*for Borland Turbo C*		95	**79**
TurboWINDOW/Pascal	*for Borland Turbo Pascal*		95	**79**
microport products			
DOSMerge	*All Varieties*		CALL	**CALL**
System V/386 Combination			799	**669**
System V/AT Combination			549	**465**
microsoft products			
Microsoft C Compiler 5	*w/CodeView*		450	**285**
Microsoft COBOL Compiler	*with COBOL Tools*		700	**439**
Microsoft Excel			495	**319**
Microsoft FORTRAN	*Optimizing Compiler*		450	**285**
Microsoft Learning DOS			50	**38**
Microsoft MACH 20		*New*	495	**329**
Microsoft Macro Assembler			150	**99**
Microsoft Mouse	*All varieties*		CALL	**CALL**
Microsoft Pascal Compiler			300	**189**
Microsoft QuickBASIC			99	**66**
Microsoft QuickC			99	**66**
Microsoft Windows			99	**66**
Microsoft Windows 386			195	**129**
Microsoft Windows Development Kit			500	**299**
Microsoft Word			450	**285**
Microsoft Works			195	**129**
periscope products			
Periscope I	*with Board*		345	**275**
Periscope II	*with NMI Breakout Switch*		175	**139**
Periscope II-X	*Software only*		145	**105**
Periscope III	*8 MHz version*		995	**795**
Periscope III	*10 MHz version*		1095	**875**
peter norton products			
Advanced Norton Utilities			150	**89**
Norton Commander			75	**55**
Norton Guides	*Specify Language*		100	**65**
Norton Utilities			100	**59**
sco products			
XENIX System V for PS/2		*New*	CALL	**CALL**
XENIX Complete System V 286			1295	**979**
XENIX Complete System V 386			1495	**1145**
other products			
ACTOR	*by Whitewater Group*		495	**419**
Brief	*by Solution Systems*		195	**CALL**
Dan Bricklin's Demo II	*by Software Garden*		195	**179**
Desqview	*from Quarterdeck*		130	**115**
DMS Resident-ASM	*by American Software Intl*		150	**139**
DMS Screen Master	*American Software Intl*	*New*	200	**185**
Finally	*by Komputerwerk*		99	**85**
GRASP	*by Paul Mace Software*	*New*	99	**85**
Heap Expander	*by The Tool Makers*	*New*	60	**55**
HI-Screen XL	*by Software*	*New*	149	**119**
Interactive EasyFlow	*by HavenTree Software*		150	**125**
Mace Utilities	*Paul Mace Software*		99	**85**
OPT-Tech Sort	*by Opt-Tech Data Proc*		149	**99**
OPTASM	*by SLR Systems*	*New, Sale*	195	**159**
Pascal-2	*by Oregon Software*		395	**289**
Peabody	*by Copia Intl, Specify Language*	*New*	100	**89**
QBase	*Relational DB by Crescent Software*		99	**89**
QuickPak	*by Crescent Software*		69	**59**
Quilt Computing Combo	*QMake & SPMS*		250	**199**
risC	*by IMSI*	*New, Sale*	80	**59**
Taskview	*by Sunny Hill Software*		80	**55**
Tmark	*by Tangent Designs*		80	**69**
Turbo GhostWriter	*by ASCII*	*New*	289	**229**
TurboHALO	*from IMSI*	*Sale*	95	**69**
Vedit Plus	*by CompuView*		185	**128**
XenoCopy-PC	*by XenoSoft*		80	**69**

CALL for Additional Products

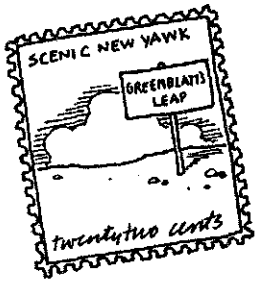
Number **146** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

from each state" would be pleased to get the query-program back that afternoon. But given an NL interface/automatic programmer, tests have shown that "inefficiency" (!) becomes a concern about 15 seconds after the request is typed.

On the issue of maintainability, it's hard to know how to respond. Would Mr. Myers consider using **byte(s[0])** instead of **length(s)** a breach? My opinion: A smarter compiler would compile these to the same code and not make library calls to get the length of a simple string variable. Similarly, Borland chose to have Turbo generate smaller code by using lib calls for evaluating **p=nil**. In situations where speed is imperative, what's wrong with substituting **((Seg(p^)** or **Ofs(p^))=0**?

If you use care, nothing that saves space or time—and compiles—is wrong. Can avoiding costly WITHs, or ordering CASE branches, be dangerous? I wrote about them because I felt they were useful and hard to abuse. On the other hand, there are things we do here that scare even me. For example, when a proc that does its own inline RET is called as the last statement in all procs that invoke it, the RET can be modified to "double-pop" back to the caller's caller. Now that's hard to maintain. But we need those clocks!



Jon Greenblatt
President
ELFSOFT
New York, NY

Democrats and Republicans

"The Second Coming—Paging and Segmentation, Multics, and OS/2" by Christian Walker (*PJ*, Volume 5.6, November/December 1987) was the best written, best explanation of segments I have ever read. It was very clear and concise.

An additional distinction must be made between hardware segments and software segments. I have nothing against software segments.

The big disadvantage of hardware segments (other than those severely limited ones in the 8088 and 80286) is the additional complexities, lower yields, and consequently higher cost of the microprocessor. At present, the relatively simple 68020 can be had for $170 while the 80386 costs as much as $500. I would rather have high speed on-chip cache memory than "big brother" segment architecture watching over me.

Unix has been getting along quite well for years running on the 68000 without hardware segments. Thousands of 68000-based Unix running workstations are humming away everyday without crashing every two hours.

This fact brings up another question. Why OS/2 anyway? Why not just Unix? Unix has far more applications than OS/2, and it, too, can run MS/DOS as a task. Why start over?

Hardware segments are the Democrats (intrusive, big brother government) of micro architecture, and a large linear address space is the Republicans (large army).

Paul Lamar

From the Mnematics PJ SIG

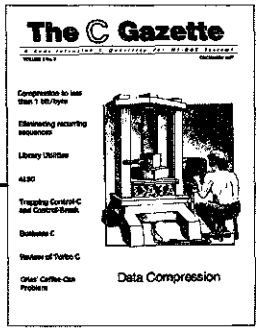
I was disappointed with Paul Barkley's "How to Get Rich in Consulting" (*PJ*, Volume 5.6). He's usually pragmatic, and the solution (highlighted in big letters) "stop charging fees on an hourly basis, and begin charging fixed fees for identifiable deliverables" is a wonderful idea. There's just one catch—the client has to go along with the idea.

While many (ex) consultants I know have sought sanctuary from the IRS as a result of Section 1706, they'd leave their jobs in a minute if there were the possibility of getting fixed-fee work and avoiding Section 1706.

You can see why Paul Barkley's flip remarks about 1706 and employment angered me and associates who are or were consul-

We have more programs than the TV guide.

The C Gazette is code and supporting text from cover-to-cover. It's the only code-intensive C publication for MS-DOS.® In-depth coverage of algorithms, detailed technical explanations, business applications, tutorials, and more. Few ads, limited reviews. If you want code, subscribe to *The C Gazette*.



___ 1 year (4 issues) $21. ___ 2 years (8 issues) $38.

We'll bill you with the first issue. If not satisfied, write cancel on the invoice, and keep the first issue *free*.

Name

Address

City *State* *Zip*

The C Gazette

A Code-Intensive C Quarterly For MS-DOS Systems

1341 Ocean Ave #257 Santa Monica, CA 90401 (213) 473-7414

Contact this advertiser directly

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DICTIONARY

A **Dictionary** is a collection of data pairs, much like an English language dictionary is a collection of term/definition pairs. The "term" entry is the key and the "definition" entry is the value of the pair, or association.

A Dictionary functions like a single-key database. This is how you create an object of class Dictionary with room for two key/value pairs:

```
Worker1  :=  new(Dictionary,  2);
```

Next fill the dictionary with strings as keys like "Name" and "Age" and the corresponding values, the string "Sam Jones" and the integer 22. You are free to mix data types.

```
Worker1["Name"]  :=  "Sam  Jones";
Worker1["Age"]  :=  22;
```

Like a database, an Actor dictionary grows when you add to it more entries than you originally specified. Below, the dictionary grows from 2 to 4 as you add two more entries, or pairs:

```
Worker1["Department"]  :=  #Engineering;
Worker1["Reviews"]  :=  #(85  73  98  94  100);
```

Mixing data types allows great flexibility. You can even incorporate the above dictionary, Worker1, as a value in an entry of another dictionary, as in Employees below:

```
Employees  :=  new(Dictionary,  10);
Employees["Sam"]  :=  Worker1;
```

Access the data in a dictionary by specifying the appropriate keys. The following statement returns the array #(85 73 98 100):

```
AWorkersReview  :=  Employees["Sam"]["Reviews"];
```

You can also retrieve data from dictionaries by enumerating over the dictionary and extracting particular entries. Here is how to create a separate "database" of employees working in the Engineering department:

```
Engineers  :=  extract(Employees,  {using(employee)
   employee["Department"]  ==  #Engineering});
```

Another feature allows you to create a new database incorporating only a subset of the original's data. You do it by enumerating over a dictionary and collecting particular data about every entry:

```
MailingList  :=  collect(Employees,  {using(employee)  employee["Address"]});
```

Dictionary objects are available in Actor as ready-made units of functionality. They can be modified or specialized for any application. However, as programmers, you are never involved in a dictionary's physical implementation or memory management.

Technical Specifications

- Runs with Microsoft Windows 1.x, 2.0 and 386.
- Pure, single-inheritance object-oriented language, incrementally compiled.
- Dynamic linking to C, Pascal, Assembler, or Fortran libraries. Pass data in C structures.
- Pascal and C-like syntax.
- Programming tools: Browser, Inspector, Debugger, File Editor.
- Full access to MS-Windows systems calls, multitasking, and DDE.
- Fast device-independent graphics: lines, shapes, icons, cursors, bitmaps, metafiles, Turtle graphics, sample control language using YACC.
- 150 classes, 1500 functions, fully extensible.
- Window styles: tiled, overlapping, popup, child, edit, dialogs. Controls: list boxes, scroll bars, buttons, check boxes.
- Data structures: stacks, arrays, queues, lists, dictionaries, sets, sorting, hashing, intervals.
- AI support: frames, symbols, dictionaries, lists, symbolic programming, functional arguments. Parsing and lexical analysis YACC compatible.
- String manipulation: substring, concat, append, insert, remove, search.
- 643-page manual includes tutorial and reference.
- No license fees. Generates stand-alone applications.
- Fastest interactive OOL available.

Prices

Actor $495 • Academic $99 • Academic site license $99 • Manuals for site license $35 • ***New!*** *Language Extension I $99 • Shipping $5 US, $25 Int'l*
Tech Support*: Level 1 (20 phone calls) $100 • Level 2 (Developer's package) $250*
Training Courses*: 201 Intro. to Actor and OOP (2 days) $395 • 202 Actor Developer's Workshop (3 days) $795 • 203 Both 201 and 202 (1 week) $995*

ACTOR®

Class of the Month

Vol. 1
No. 1

Actor: an interactive object-oriented programming environment running under Microsoft Windows.

Class: a template for data structures, or objects, with similar properties. These objects are tested, refined and reusable units of functionality you can snap into your Actor programs at any time.

The Whitewater Group®
Technology Innovation Center
906 University Place
Evanston, Illinois 60201

For more information call: (312) 491-2370

Actor is a registered trademark of The Whitewater Group, Inc.

Number 73 on the Reader Service Card.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

tants. Gee, Paul, it's great that you've got the problem licked, but your readers will turn the page if you keep up that level of arrogance.

Maybe it's not the same all over the country, but here in New York City, the problem is that all the potential clients (with money to pay, that is) want programmer meat. It's the only way they know how to do business. The consultant is not in a position to educate ignorant and stubborn clients about such issues as productivity. Neurotic executives and managers (so inept that they don't trust their employees and/or consultants) feel that if the body isn't in place and in full view, the work isn't getting done. Somehow, the logic of fixed-price contracts eludes these managers, who aren't in any case held fiscally accountable.

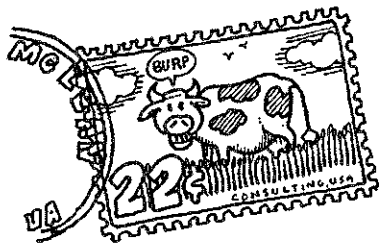
Where does Paul live, anyway? After reading the last article, I've come to wonder if it's in an ivory tower (even if of his own construction).

Fred Salamander
New York, NY

Herbivores and Carnivores—
Paul Barkley Replies

There are many kinds of consulting, and each has its risks and rewards. My clients are primarily small businesses, or single departments within large businesses. The program deliverables are always small and identifiable because I generally don't accept tasks that are ill defined and that tend to lead to perpetual employment. I don't code until I write a system design and get it approved.

Many other consultants, though, work on site for large companies and get paid by the hour, very much like employees. (One consultant pointed out to me that if you do mainframe work, it's hard to justify spending 3 or 5 million dollars for a machine to put in your basement—you pretty much have to work on site.) This type of hourly work is very far removed from the kind of consulting I do. These hourly people are sometimes called "contract programmers" to distinguish them from "consultants," but I don't want to step on any more toes by insisting that only people who work the way I do can call themselves consultants.



Reader Salamander is correct—companies that hire "programmer meat" are sure not going to hire anyone on a fixed fee basis, and thus my suggestion is not feasible for anyone in that line of work. This "meat" attitude, however, nicely illustrates why the IRS took a dim view of people who work like employees but claim to be consultants; the IRS didn't see any difference between the two groups.

I'm sorry if I seemed flippant about Section 1706. If it were my life that were turned upside down, I would be unhappy, too. This legislation did not affect a single "consultant" that I know, but it sure wiped out the "contract programmers." Those affected have been very vocal.

The important thing is to recognize the nature of the work that you do, or want to do, and how it differs from the kind of work practiced by others. Then, maximize your financial and mental returns by adjusting your business strategy. If you feel that all the clients you know are ignorant and neurotic, you simply need to identify a different market segment. We have all kinds of clients here in Washington, too. Try smaller clients, but be prepared for small budgets—they want small systems, fast results, and a small bill. Hey, if consulting were easy, everyone would be doing it, huh?

One last thought: If making money is your only goal, you may find that accepting "contract programming" jobs, even as a temporary employee, will pay better year after year than "consulting" because of the erratic (and thus less than full time) nature of independent consulting. It just depends on your particular practice. It's the herbivore that

If you can find better sort/merge software, buy it!

Sortex:® A flexible, general purpose sort/merge facility which provides you with the ultimate in performance and reliability

- Sortex can be run from the DOS prompt, batch file, or applications written in Assembler, BASIC, COBOL, FORTRAN, Pascal, or C.
- Sortex can accept your requirements through the full screen menu/panel facility, control file, or command line.
- Sortex can process most of the commonly used file and data types.
- Sortex can produce output files with key, address, index, tag, or full records.
- Sortex imposes no practical limitations. Files with two billion characters, records & fields with thirty two thousand characters each, and unlimited number of keys can be handled by Sortex.
- Sortex allows records to be chosen selectively for processing.
- sortex can sort your records in ascending/descending order on any number of keys.
- Sortex can merge your sorted record sequences into one sequence on any number of keys.
- Sortex preserves the input order of records on the output file if all their keys are identical.
- Sortex allows keys to overlap, or be contained within other keys. They need not be contiguous.
- Sortex can process files that contain standard & nonstandard field delimiters, record separators, & file terminators.

30-Day Money-Back Guarantee Only $99.95

Systemat CORPORATION

1915 W. Orangewood Avenue
Orange, California 92668
Telephone: 714 978 0480 Telex: 910 250 0674

Number 171 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

usually has a full belly from grazing. The carnivore often goes hungry while stalking his next meal.

Paul Barkley
McLean, VA

INT 2Eh Complications

Oh, oh, oh! I just read John Newlin's very interesting article, "The Shell Game" in the November/December 1987, issue of *PJ*. He made a big mistake choosing INT 2Eh for his EXEC function. Indeed, this interrupt does provide a quick, down and dirty way to execute another program. But, it has one major flaw. It is nonre-entrant. If the program that is executed also makes use of INT 2Eh (and why not—if it is good enough for JN, it might be good enough for other programmers), you will never return to the shell. Nested calls to INT 2Eh will return at most one level via the EXIT function.

If you follow INT 2Eh in a debugger, you will see why. (JN, why didn't you do this? If you use an undocumented call/int, you ought to try to document it.) INT 2Eh does some house cleaning, sets the registers for a call to (you guessed it) Function 4Bh, saves the current SS:SP and the return address, and branches to the execute function code. A second call to INT 2Eh therefore overwrites the return address of the first call.

This complication is in addition to those connected with Function 4Bh.

Microsoft's exact purpose in designing INT2Eh is unclear to me. In some DOS versions it is used to execute BATCH files. This is why they cannot be nested unless a new command interpreter is loaded to execute them. This avoids the re-entry problem.

Some time ago I wrote an execute function for the purpose of providing DOS access to programs that do not do so themselves. For example, you might want to call your text editor from inside SC3. This routine is pretty near crash proof. It is by no means as short as the down and dirty INT 2Eh—it is perhaps 3–4 pages of assembler code.

Larry Smith
Friedlands, West Germany

John Newlin Replies

Mr. Smith is correct. DOS interrupt 2Eh is not re-entrant and therefore is not the best choice for executing other DOS programs from the Shell. I should have instead used the Marshal Pascal intrinsic **LoadProg** procedure, which accepts program name and command tail strings as parameters and passes these to DOS via function 4Bh of interrupt 21h. I have used interrupt 2Eh for so long that I added it to the Shell program as a matter of course and without due technical consideration.

John J. Newlin
San Diego, CA ♦

FINALLY!

EASYFLOW

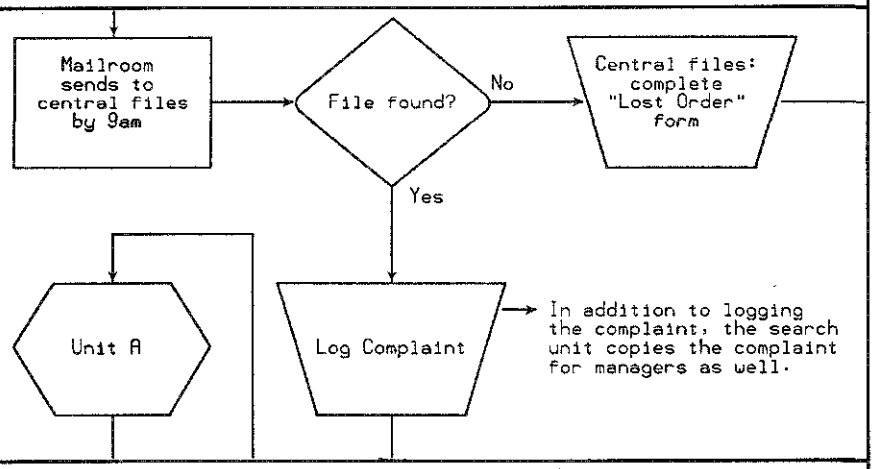
An on-screen flowchart processor that knows about flowcharts - not just another "screen draw" program that makes you do most of the work. EASYFLOW is a powerful full-screen graphics program dedicated to flowcharts and organization charts. With it you can quickly compose charts. More important, you can easily modify charts so they are always up to date.

- **Automatic:** Fully automatic text centering within shapes, both horizontally and vertically. Fully automatic line routing & re-routing.
- **Fast:** Written in assembly language for speed.
- **Large:** Charts up to 417 columns wide by 225 lines high. Chart too large for your printer? EASYFLOW automatically breaks the chart up & prints it in page size pieces.
- **Standard:** All standard flowcharting shapes included. Custom shapes can be ordered.
- **User friendly:** Don't take our word for it. PC Magazine* says "EASYFLOW lives up to its name. It's hard to imagine any easier and more flexible way to produce basic and even complex flowcharts".
- **Mouse:** Optional but fully supported.
- **It prints:** On most popular matrix printers including IBM, Epson, Toshiba, HP LaserJet, LaserJet-Plus and many others.
- **It plots:** On HP7440, 7475, 7550, 7570, 7585B and compatible plotters.
- **It works:** We are contractually prevented from mentioning the name of the "big eight" accounting firm that purchased a world-wide site license, but we can tell you that they spent months evaluating all available flowcharting packages before choosing EASYFLOW.
- **Rush delivery:** Order by noon today (eastern time) and we'll have it to you by courier tomorrow**. Rush delivery charge is $15.00 (instead of $2.00) and is available only in USA & Canada.
- **Documented:** 100 page manual plus over 150 screens of context sensitive help.

EASYFLOW works on IBM PC's, IBM PS/2 and compatibles. Requires 384 K memory, DOS 2.0 or higher and an IBM CGA/EGA/VGA or Hercules monochrome compatible adapter card.

Order direct for only **$149.95** + $2.00 S&H (USA/Canada), $10.00 (foreign). Payment by M.O., cheque, VISA, Mastercard or Company PO.

* March 10, 1987 issue, page 278.
** Rush orders are shipped by Purolator Courier and normally arrive the next business day to most locations. Remote destinations take longer.


Mailroom sends to central files by 9am
File found?
No
Central files: complete "Lost Order" form
Yes
Unit A
Log Complaint
In addition to logging the complaint, the search unit copies the complaint for managers as well.

The chart fragment above was produced on an HP LaserJet-Plus and is actual size and unretouched. Publication quality charts like this can be produced using only minutes of preparation time and seconds of print time.

HavenTree Software Limited
PO Box 1093-V
Thousand Island Park, NY 13692

Order Desk: 1-800-267-0668
Information: (613) 544-6035 ext 31
Telefax(G3): (613) 544-9632

Number 154 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

VEDIT PLUS

#1 PROGRAMMABLE EDITOR

Call 1-800-45-VEDIT for FREE Fully Functional Demo Disk

Stunning speed. Unmatched performance. Total flexibility. Simple and intuitive operation. The newest VEDIT PLUS easily satisfies the most demanding computer professional.

Try a Dazzling Demo Yourself.

The free demo disk is fully functional—you can try all features yourself. Best, the demo includes a dazzling menu-driven tutorial—you experiment in one window while another gives instructions.

The powerful "macro" programming language helps you eliminate repetitive editing tasks. The impressive demo/tutorial is written entirely as a "macro"—it shows that no other editor's "macro" language even comes close. And VEDIT PLUS is only 40K in size.

Go ahead. Call for your free demo today, You'll see why VEDIT PLUS has been the #1 choice of programmers, writers and engineers since 1980.

Only VEDIT PLUS is this Flexible.

The installation lets you pick from closely emulating the keyboard layout of Word Perfect, WordStar and others. Or you can easily create your own layout and even your own editing functions. Supports any screen size—you pick screen colors and attributes.

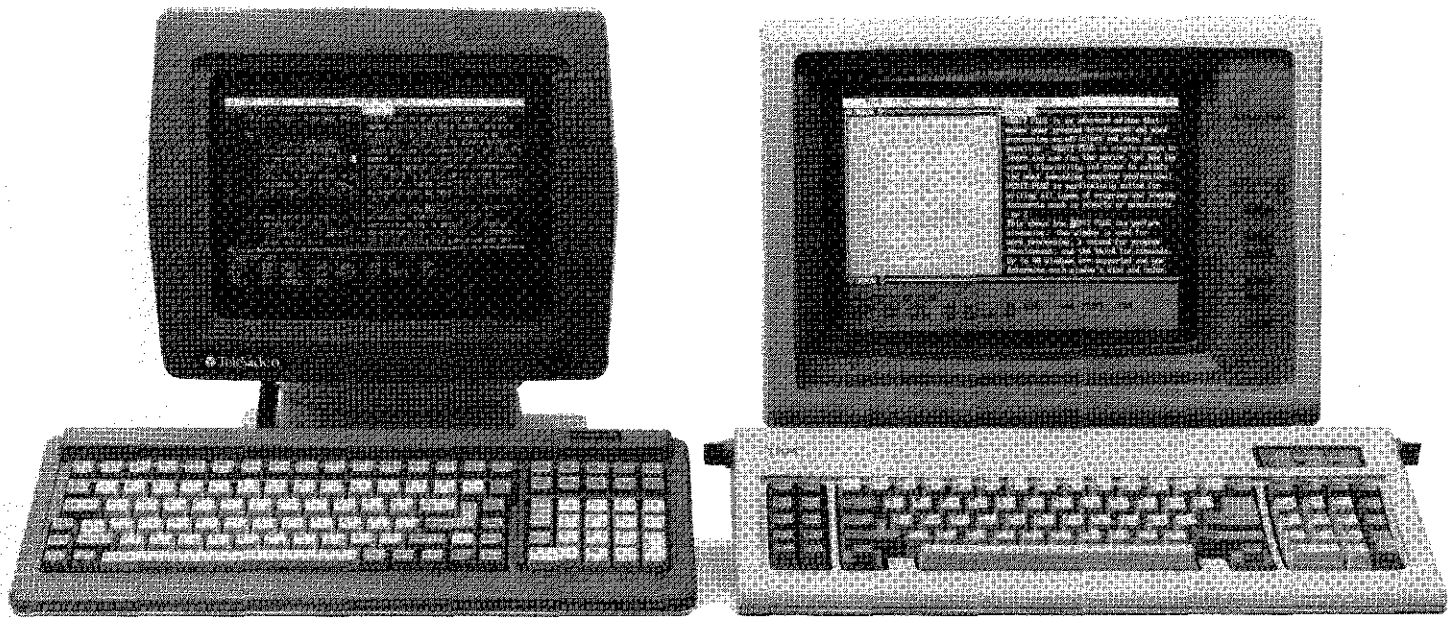
Supports the IBM PC, XT, AT and PS/2. Also supports MultiLink, PC-MOS/386, Concurrent DOS and most networks. Also available for MS-DOS, FlexOS (protected mode), CP/M-86 and CP/M. (Yes, we support windows on most CRT terminals, including CRTs connected to an IBM PC.) Order direct or from your dealer. $185.

Special: VEDIT (single file, no windows) for CP/M—$49.

- **Fully Network Compatible**
- **Call for XENIX-286 version**
- **30 Day Money-back guarantee**

Compare Features and Speed

	BRIEF	Norton Editor	PMATE	VEDIT PLUS
'Off the cuff' macros	No	No	Yes	Yes
Built-in macros	Yes	No	Yes	Yes
Keystroke macros	Only 1	No	No	Unlimited
Multiple file editing	20+	2	No	20+
Windows	20+	2	No	20+
Macro execution window	No	No	No	Yes
Pop-up menus	No	No	No	Yes
Execute DOS commands	Yes	Yes	Yes	Yes
Automatic processing of Compiler errors	Yes	No	No	Yes
"Cut and paste" buffers	1	1	1	36
Undo line changes	Yes	No	No	Yes
Paragraph justification	No	No	No	Yes
Convert to/from WordStar	No	No	No	Yes
On-line calculator	No	No	No	Yes
Configurable Keyboard	Hard	No	Hard	Easy
43 line EGA support	Yes	No	No	Yes
Manual size/index	250/No	42/no	469/Yes	380/Yes
Benchmarks in 120K File:				
2000 replacements	1:15 min	34 sec	1:07 min	6 sec
Pattern matching search	20 sec	Cannot	Cannot	2 sec
Pattern matching replace	2:40 min	Cannot	Cannot	11 sec



VEDIT and CompuView are registered trademarks of CompuView Products, Inc. BRIEF is a trademark of UnderWare, Inc. PMATE is a trademark of Phoenix Technologies Ltd. Norton Editor is a trademark of Peter Norton Computing Inc. MultiLink and PC-MOS/386 are trademarks of the The Software Link, Inc. CP/M and FlexOS are trademarks of Digital Research. MS-DOS is a trademark of Microsoft.

*Also available for TI Professional, Tandy 2000, DEC Rainbow, Wyse WY700 and others.
*Demo disk is fully functional, but does not readily write large files.

CompuView

1955 Pauline Blvd., Ann Arbor, MI 48103
(313) 996-1299, TELEX 701821

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

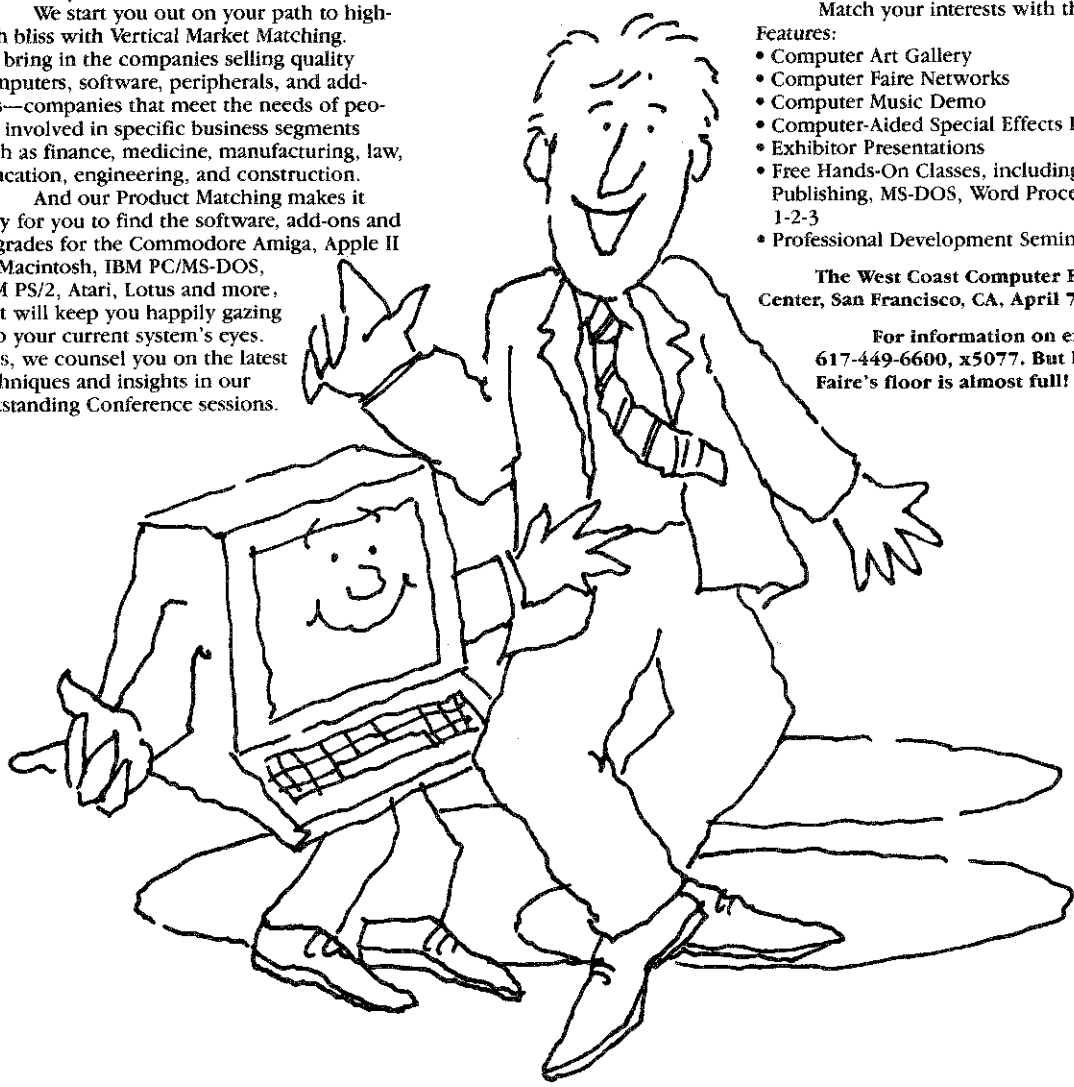
PROGRAMMER'S JOURNAL
WILL BE AT THE FAIRE!
SEE US IN BOOTH #439

The West Coast Computer Faire announces the first Computer Matchmaking Service.

You won't have to depend on fate at the 13th West Coast Computer Faire to find the products and services that are the perfect match for your needs.

We start you out on your path to high-tech bliss with Vertical Market Matching. We bring in the companies selling quality computers, software, peripherals, and add-ons—companies that meet the needs of people involved in specific business segments such as finance, medicine, manufacturing, law, education, engineering, and construction.

And our Product Matching makes it easy for you to find the software, add-ons and upgrades for the Commodore Amiga, Apple II or Macintosh, IBM PC/MS-DOS, IBM PS/2, Atari, Lotus and more, that will keep you happily gazing into your current system's eyes. Plus, we counsel you on the latest techniques and insights in our outstanding Conference sessions.



The West Coast Computer Faire has made and will make more matches than any another computer show. It's time we made the perfect match for you.

Match your interests with these Faire Features:

- Computer Art Gallery
- Computer Faire Networks
- Computer Music Demo
- Computer-Aided Special Effects Demo
- Exhibitor Presentations
- Free Hands-On Classes, including Desktop Publishing, MS-DOS, Word Processing, Lotus 1-2-3
- Professional Development Seminars

The West Coast Computer Faire, Moscone Center, San Francisco, CA, April 7-10, 1988

For information on exhibiting, call 617-449-6600, x5077. But hurry — the Faire's floor is almost full!

Register early and save $5!

Fill out this coupon and mail with your check(s), for $15.00 for each registrant, postmarked by March 17th, 1988. Include the names and addresses of registrants for whom you are enclosing a check. (Photocopy coupon for additional registrants.)

PJ 114

Name _____ Title _____

Company _____

Address _____

City _____ State _____ Zip _____

Phone (_____)_____

Four day conference and exhibits $15.00 in advance. $20.00 at the door. Make check payable to "The West Coast Computer Faire." Mail to: Attendee Registration Department, The West Coast Computer Faire, 300 First Avenue, Needham, MA 02194. Advanced registrations accepted only with full payment and each registrant's name and address. Tickets will be mailed to each individual registrant separately.

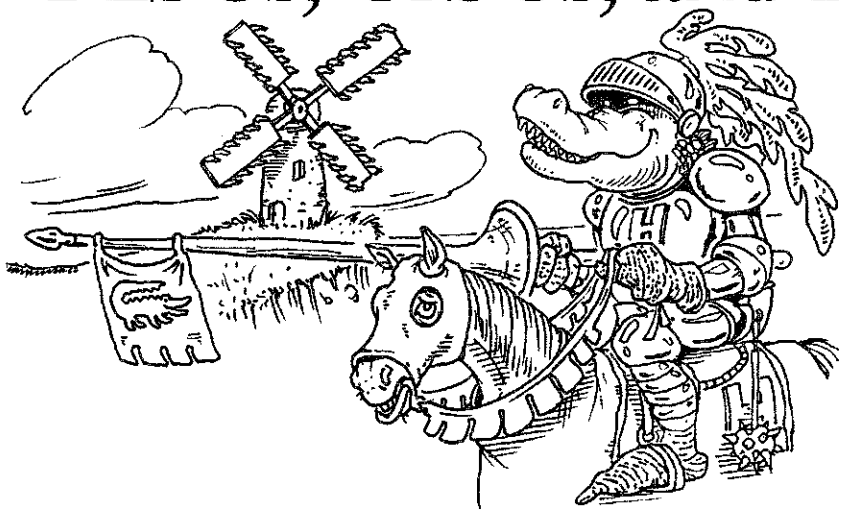
THE 13th WEST COAST COMPUTER FAIRE

April 7–10, 1988, Moscone Center, San Francisco, CA

Number 58 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

RISCs, CISCs, and Fabs



by
Hal W. Hardenbergh

You are probably wondering why a 680x0 enthusiast is writing for a PC-related mag. The publisher threatened to make me pay *for my subscription to this rag if I didn't!*

I've been reading lately that the RISC-CISC controversy is over and RISC has won. Now, you'd expect the folks flogging RISC chips or boards or systems to say that, but some neutral observers are signing up with the RISC faction. Stan Baker of *EET*. John Dvorak of *PC*. (For those of you who have just returned from an extended vacation on the planet Zorn, RISC means Reduced Instruction Set Computer, and CISC means Complex Instruction Set Computer. RISCs use but a single clock per instruction, they have lots of registers, the instructions are always one 32-bit word aligned with the 32-bit bus, etc.)

Are RISC chips better than CISC chips? Good question. It is, however, also the *wrong* question, honest! For starters, we can't say whether RISCs are superior to CISCs, or vice versa. All we can do is compare *implementations* of RISC and CISC. But that's just one problem. How can we tell which chip is a RISC and which is a CISC?

Take the Transputer, for instance. It has *three* registers! Its instructions are one or more bytes in length, which means they can't possibly be aligned with that 32-bit bus! And the latest Transputer not only adds floating point instructions, but also adds—get this—specialized instructions for moving bit-mapped windows and such. *This is a RISC?* Understand, not even the 680x0 or 386, which are universally considered CISC machines, have floating point instructions or bit-mapped window movement instructions. And the 680x0 and 386 both have more—a lot more—than three registers. In other words, *if the Transputer is a RISC, then there ain't no such thing as a CISC!*

The Transputer simply does not pay its own way . . . the Clipper seems to be sailing under a curse . . . and nobody will touch the WE32 series with a ten-foot pole.

Remember, we're talking about microprocessors here. *Here are the correct questions:* Which microprocessor implementations will prove commercially successful in the personal computer marketplace? Of the successful chips, which will prove *most* successful?

The first popular 32-bit microprocessor was the 68020, which is currently being built at a rate of about 1 million per year, and that rate is accelerating. The next-generation part, the 68030, is now available at the chip (but not system) level. The second popular 32-bit chip was the Intel 386, which is also being built at the rate of 1 million per year, and that rate is accelerating. The 68020 and 386 are popular mostly because they are each software-compatible with previous-generation micros, while running that compatible software faster.

Here are some contending merchant 32-bit chips: the Nat Semi 32x32, the Zilog Z-80000, the Acorn ARM, and the Transputer. The T.I. 34010 graphics chip is really a general purpose RISC chip, is readily available, and is available in a 50 MHz (!) version. AMD is sampling the 29000 *and* the 29300. Motorola will have officially announced the 78000 RISC chip by the time you read this. Intel has leaked that it has its own RISC chip in the works, as well as a 486.

And then there are the proprietary chips: the Sun SPARC, the Mips, Inc., RISC chip, the Intergraph (nee Fairchild) Clipper, the chip used in the IBM RT, the HP SPECTRUM chip, the MicroVAX chips . . . am I leaving some out? Oh, yes: Edge. More on Edge later.

The RISC-CISC controversy is over, and RISC has won? Really? Does that mean that 68020 and 386 sales are going to fall 'way off, or does that mean that some RISC chip is going to (soon?) be made at the rate of *ten* million chips per year? (Fujitsu is making the SPARC right now at the rate of about 1000 per month and is projecting production of 10,000 per month at the end of 1988.) If commercial success or failure is not related to production rate, RISC might win.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Contenders:

Sun's SPARC chip is advertised to have 10 MIPS performance, but averages about 6.5 MIPS. Although tie-ins with AT&T and Xerox have been announced, Sun is the only outfit shipping the SPARC. It is most unlikely that the SPARC chip will ever make an appearance in the personal computer arena. Oh, yes: Sun doesn't make the SPARC. Fujitsu makes it.

The Nat Semi 32532, which was developed by a team of chip designers in Israel, is now being sampled. It's advertised to out-perform the 68020 and 386, but I gave up believing performance claims on the 32000 series family long ago. Besides, the 32532 is contemporary with the 68030, not the 020, and should be compared with the 030. The 32032 has sunk without a trace, and the 32332 apparently shares the fate of the 80186.

The Zilog Z80000, which was announced with such fanfare back in June of 1983, is now available. A world-beater it ain't, to put it politely. It is currently available with an 8 MHz clock, for instance. Yeh, *tomorrow* it will be faster, but for Zilog tomorrow never seems to arrive.

The Acorn ARM (Acorn RISC Machine) is a 32-bit version of the venerable 6502 microprocessor. It's used in the Acorn Archimedes computer, which you read about recently in *BYTE* magazine. A pretty good low-budget micro. Acorn doesn't make it—VLSI Technology, Inc., does.

The Transputer has now been around for awhile, and nobody is getting rich making Transputer-based systems. On the other hand, Floating Point Systems, Inc., has reported substantial losses and changed management a couple of times because it dumped a lot of money into a Transputer-based computer (the Series T). The British government got tired of dumping money into Inmos to support the Transputer and sold out; the British buyer (Thorn-EMI) has gotten tired of dumping money (would you believe over $400 million?) into Inmos to support the Transputer and is now trying to sell out, perhaps to Japan. The Transputer simply does not pay its own way. And like I said, it's been around for awhile.

The T.I. graphics chip, the 34010, is a general-purpose RISC chip with some specialized CRT-support circuits and instructions. It's cheap, available (100,000 have been shipped), and runs at up to 50 MHz (!).

The Clipper seems to be sailing under a curse. Fairchild funded its development but got auctioned off to Nat Semi, which already had a line of 32-bit microprocessors. So Intergraph, a large CAD/CAE company that had developed a design based on the Clipper, bought the Clipper for a munificent $10 million. Intergraph is supposedly going to offer the chip set to the general public. Shipments to date have been 25 MHz rather than the advertised 33 MHz; 33 MHz parts will begin volume shipments early in 1988, and a 50 MHz version is said to be in the works. Oh, yes: The Clipper comes as a set of chips on a board (for $1500), so it is not really a microprocessor. It does not seem suitable for the personal computer market-place. AMD is now sampling the 29000, which a lot of people seem to regard as "son of 2901," referring to the ubiquitous 2901 bit-slice chip of microprogramming fame. Wrong: The 29300 chip family is the next-generation "2901," based on microprogramming, and hence a CISC architecture. If there is such a thing as a true-blue RISC chip, the 29000 is one of them. It's too early to judge the future of this chip.

There is a severe shortage of 1.5 micron fabs, and we need a better word than "severe" to describe the shortage of 1 micron fabs.

Novix makes a dedicated FORTH chip, but it's a 16-bitter, and we're talking about 32-bit chips here. Aren't we?

The Mips, Inc., RISC chip was originally going to be proprietary, then it was going to be sold on the open market, and now it's proprietary again. Obviously, no proprietary chip is going to make it big in the personal computer marketplace. The Mips chip (and its floating-point chip) *may* be the best-performing chip-based microcomputer now actually being shipped as a board-level product.

So far we haven't mentioned two 32-bit CISC chips, the NEC V60/70 and the AT&T WE32 family. Unlike the NEC V20/25/30/50, the V60/70 is *not* based on the Intel architecture. NEC is targeting the V60/70 at embedded applications, and nobody will touch the WE32 series with a ten-foot pole. Well, almost nobody.

There are *lots* of proprietary 32-bit chips. It seems pointless to mention them because you can't buy one (as a chip) and none of them will ever be competitive in the personal computer market.

Where's the Software?

Let's face it, it takes a lot of software to be successful these days. The fastest 32-bit chip now shipping and that has lots of software support is the Motorola 68030. And just as Intel has admitted that it is working on the 486, Motorola has admitted that it is working on the 68040. I can't see any other chip, whether RISC or CISC, beating out the Intel and Motorola lines in the personal computer marketplace.

It takes more than raw speed to be a success in the personal computer marketplace these days. I have some personal experience with that. I tried to sell DBASIC, a very fast, incrementally compiled BASIC, for the Atari ST. DBASIC has failed miserably—it sold 70 copies at $40 each in four months. Naturally, I've given up trying to sell DBASIC. You've read about wonderful QuickBASIC 4.0, which is wonderfully incrementally compiled? Well, DBASIC was/is incrementally compiled, exactly as is QB 4.0. But DBASIC, with a command-line interface just like

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MetaWINDOW

Power Graphics for your PC!

PC TECH JOURNAL

"Product of the Month"
"... a technological tour de force for fast PC graphics."

NO ROYALTIES!

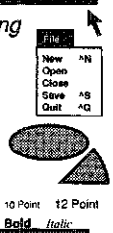
MetaWINDOW is ... pack...

NEW! - QuickWINDOW/C
All the features of MetaWINDOW for Microsoft Quick/C! - $95*
...features and ...window managers.

Unparalleled Performance!

MetaWINDOW provides an expanded set of graphic drawing functions, plus the added functionality and performance required for designing multi-window desktop applications.

- *auto-cursor tracking*
- *pull-down menus*
- *pop-up windows*
- *comprehensive graphic functions*
- *multiple fonts*



Enhanced Features!

- *Display multiple bitmap or "filled-outline" fonts.*
- *Face fonts for bold, italic, underline or strike-out stylings.*
- *Full "RasterOp" transfer functions for writing, erasing, rubberbanding or dragging: lines, text, icons, bit images and complex objects.*
- *Create pop-up menus, windows and icons.*
- *Supports IBM's new PS/2 VGA and MCGA graphics.*

MetaWINDOW comes complete with language bindings for 20 popular C, Pascal and Fortran compilers, plus dynamic runtime support for over 50 graphics adaptors and input devices.

MetaWINDOW
Advanced Graphics Toolkit
*4 disks, 3 260 page manuals - $195**

NEW! - TurboWINDOW/Pascal
*All the features of MetaWINDOW for Borland Turbo Pascal Ver. 4! - $ 95**
*Plus $5.00 shipping and handling

TO ORDER CALL 1-800-332-1550
For information or in CA call 408 438-1550

METAGRAPHICS
SOFTWARE CORPORATION
269 Mount Hermon Road
Scotts Valley, CA 95066

Number 143 on the Reader Service Card

QB 4.0's, tried to sell to folks who use their computers as toys. ST folk use mice, windows, icons, and joy-sticks to play their games, so DBASIC failed. (Too, folks who spend $500 on a personal computer, and who do not use it for business, don't want to spend much money on software.)

Similarly, there is a lot of artistic merit in some of the 32-bit contenders in the microprocessor world, whether we happen to call them RISCs or CISCs. But artistic merit does not pay the bills. (DBASIC may be the Transputer of personal computer languages.) But what the heck, you can't succeed if you don't give it a try. I don't regret giving DBASIC a try, and I assume that none of the also-rans in the 32-bit chip field regret it either.

Unmultiplys/Undivides:

All of you remember the 8080, 6800, and 6502. Those parts were slow, partly because they lacked multiply and divide instructions. Well, guess what the Acorn ARM, the Sun SPARC, the HP SPECTRUM (chip version), and the AMD 29000 have in common? None of them has multiply or divide instructions . . . and they are all *high performance* 32-bit RISC chips??

The RISC concept has been around for a long time. The CDC 6600, the first of the high-performance scientific mainframes, was a RISC machine. It was designed by Seymour Cray, and Cray's machines have been RISC-like ever since. I remember the 8x300, which was a contemporary of the 8080/6800/6502. It was fast—and it took a heck of a lot of instructions to get anything useful done, just like today's RISCs.

I just read about yet another RISC paper tiger, this one a 17 MIPS machine from Germany being developed on a $3.2 million budget. The original Acorn ARM was probably developed for less than that. It is not terribly difficult these days to develop one's own CPU design since you no longer have to own a chip factory to do so. (Acorn, Sun, and Mips, Inc., all have their chips built at foundries—as does Intergraph for the Clipper.) Designing your own CPU chip is the ultimate NIH (Not Invented Here).

So now we have, for the first time, a surfeit of CPUs? Wrong. In 1975 there were the 8080, 6800, 6502, the 8x300 just mentioned, the Signetics 2650, and the Fairchild F8, plus the Nat Semi IMP-16 and PACE, the T.I. 9900, and a General Instrument chip—all 16 bitters! NIH has been around for awhile . . .

SemWare Announces QEdit Advanced the "quick editor"

QEdit is one of the smallest, fastest, multi-file editors available for IBM compatible computers and has many features to enhance your productivity.

- **Lightning fast, especially in file loading and saving, screen display, cursor movement, and global searching and replacing.**
- **Edit multiple files, as many and as large as available memory allows. QEdit can use all available memory, up to 640KB. Ideal for multi-file languages and all those .PRG files.**
- **Windowing allows up to 7 files or 7 views of the same file to be seen at once.**
- **You do not have to "learn" another editor with QEdit. QEdit's commands can be configured to any keys you like.**
- **Screen colors, tab settings, default file type and many other options can be configured the way YOU like it.**
- **Up to 99 named scratch buffers may be created and edited. Commands are available for easy cut and paste operations.**
- **Invoke a DOS "shell" or execute DOS commands directly from QEdit.**
- **Both line and stream block marking are supported.**
- **Keyboard macros allow automation of commonly used command sequences or text entry.**
- **Commands to shift an entire block left or right, to adjust indentation levels when using structured languages.**
- **Garbage stack provides restoration of the last "n" (user-settable) deletions.**

Introductory Price
49.95
Mastercharge & Visa Accepted
Add $3.00 Shipping
Georgia Residents please include 5% sales tax

QEdit Advanced requires an IBM PC/XT/AT or Compatible, IBM PS/2 30 - 80, 128KB of available memory. Please specify disk format required.

To Order: Heritage Papers, Danielsville, GA 30633
Phone Orders: (404) 789-3226

Number 148 on the Reader Service Card

PRODUCED 2001 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

What's the fastest board-level CPU product currently being shipped? I said it may be the Mips board. But maybe not. A company called Edge shipped $6 million's worth of Edge 1000s in 1987, and the Edge 1000 comes in two flavors—a single-CPU board running at 6 MIPS and a dual-CPU running at 11 MIPS. The Mips, Inc., board runs at 10 MIPS . . . and Edge is getting ready to ship a new line of boards with a 16 MIPS single-CPU and a 55 MIPS quad-CPU version!

Edge? Who is Edge? Edge makes CPUs out of several VLSI chips that emulate the Motorola 680x0 series—that is, they execute exactly the same instruction set. So how can they run faster? Well, by throwing more silicon at the problem, they execute most instructions using fewer clock cycles than a Motorola chip. If you have some Motorola software that can't run fast enough on a 68020, you try the 68030. If it still isn't fast enough, you can either wait at least two years for the 68040 or you can buy a board from Edge now. Obviously, Edge CPUs are CISCs. And they have a substantial software base.

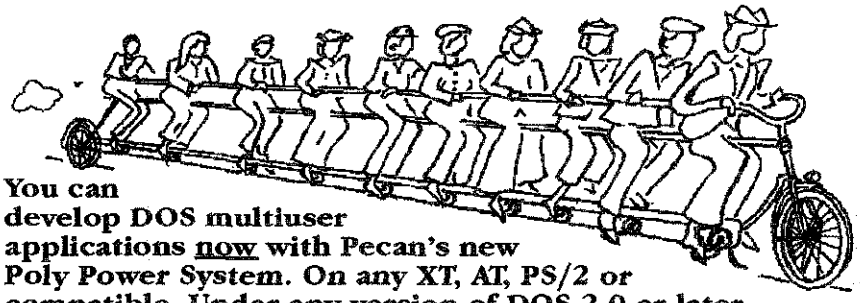
Them Fabulous Fabs:

It is now possible for anyone who has access to a 1 micron chip foundry to make a CPU that will outrun standard DRAM memory, even using a 32-bit bus. So what we're seeing is a lot of money being thrown at high-speed cache memories and their ancillary logic. It is this fact that causes me to lean towards CISC architecture as the best for future high-performance CPUs. CISC machines need to fetch fewer instructions than RISC machines do to perform the same amount of useful work. That means that either a slower memory interface can be used to perform useful work at the same rate, or that CISC machines will run faster if the memory interface ultimately proves to be the limiting factor for high performance. And I think we are rapidly approaching the point where memory speed limits the CPU (if we aren't there already). Right now you can see a lot of $250 CPUs tied to $2500 caches in an attempt to lick this problem . . .

And everybody has access to a 1 micron chip foundry, right? Well, no.

It's common knowledge that the chip industry has an overcapacity. Maybe. There is a *substantial* overcapacity of 3

MULTIUSER DOS NOW!



You can develop DOS multiuser applications now with Pecan's new Poly Power System. On any XT, AT, PS/2 or compatible. Under any version of DOS 2.0 or later.

All you need is Pecan's Poly Power System, one or more of our compilers: UCSD Pascal, Modula-2, FORTRAN-77, BASIC, Assembler (C is coming soon), and inexpensive dumb terminals on serial ports.

You can even have multiple programmers develop on a single machine.

Poly Power System, $399.95. Compilers, $79.95 each. Call us for full details and to place your order.

PECAN
The UCSD Pascal Company
1-800-63-PECAN

Pecan Software Systems, Inc.
1410 39th Street
Brooklyn, NY 11218
718-851-3100

Pecan Software Europe, Ltd.
MGM House, Oakfield Grove
Clifton, Bristol, BS8 2BN, England
0272 733633

Number 71 on the Reader Service Card

MISOSYS

SOFTWARE FOR YOUR P.C.

Order Now 800-MISOSYS (647-6797)

ED/ASM-86™

- 80X86 Assembly Language development

An integrated editor, assembler, linker, and debugger for 8086, 80186, 80286, and 8087 assembly and disassembly. All features are in a single EXE file.

- a full-featured line EDITOR with intra-line editing, block move and copy, partial save and load.
- a full-screen turbo-type editor is supported on CGA-equipped systems.
- the ASSEMBLER supports structured coding; can directly generate a COM or EXE file; assembly to memory for faster debugging.
- The DEBUGGER includes symbolic disassembly of arbitrary code to disk.

ED/ASM-86 [M-86-030] . . $124.95 + $5 S&H

RATFOR-86™

- Rational FORTRAN translator

A professional implementation of RATional FORtran by James Beard, Ph.D. Provides structure and greater portability to FORTRAN programs to reduce your programming time and effort dramatically over that required when writing directly in FORTRAN. Includes tutorial reference manual.

- compiler has extensive macro support
- "arith" macro for binary arithmetic
- short form READ and PRINT
- enhanced switch case support
- Requires a FORTRAN compiler

RATFOR-86 [M-86-73] . . . $99.95 + $5 S&H

MISOSYS Inc.
P.O. Box 239
Sterling, VA 22170-0239
703-450-4181 MC, VISA, CHOICE
800-MISOSYS/647-6797

VA residents add 4½% sales tax.
S&H: Canada add $1;
Foreign use S&H times 3

Number 121 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

micron fabs (foundries). Supply and demand seem balanced for 2 micron fabs. There is a severe shortage of 1.5 micron fabs. And we need a better word than "severe" to describe the shortage of 1 micron fabs. Why? Money, of course.

A 1 micron facility is very, very, very expensive. It also costs a lot. Why do you think the Zilog Z80000 is so slow? Partly because Exxon got tired of how much money Zilog had lost and declined to spring for a 1 micron fab. Fujitsu made the Fairchild Clipper because Schlumberger got tired of pouring money into Fairchild. Why isn't the Transputer faster? Well, parent Thorn has already put over $400 million in the pot, so Inmos doesn't have a 1 micron fab. The Motorola 68030 is built using 1.2 micron design rules. The Intel 386, Nat Semi 32532, and AT&T WE32 use 1.5 micron rules. The Acorn ARM, Zilog Z80000, and Clipper use 2 microns. And the AMD 29000 is built using 1.2 micron "effective" design rules—meaning 1.5 to 1.6 microns. (All figures are for *current* production.)

You see, small design rules are now so popular that folks are beginning to cheat. Sometimes the cheating is specifically identified by the qualifier "effective" and sometimes not. Beware! Not all 1 micron design rules are equal!

Bird in the Hand, Pie in the Sky:
Wait, we're not done yet. Question: Is a 12 MIPS chip faster than a 4 MIPS chip— if we specify that those are VAX MIPS in each case, not phony RISC MIPS? Answer: Not necessarily.

You see, CPU performance has been increasing by a factor of 2 every two years—which means an increase of 32 in ten years, and an increase of 1000 in twenty years. It has been about 17 years since the microprocessor was invented.

Take the 68020, which saw first silicon in Aug '85. Nat Semi brags that its 30 MHz 32532 is faster than the 68020. Problem: Nat Semi is now making 20 MHz chips, not 30 MHz. And it just started making those chips, over two years after Motorola started making the 68020! And there is always a brand-new paper tiger, like the German "chip" (actually chip design in progress) I just read about. If that ever becomes real, it will be at least two years hence, and by that time it will have to buck heads with the 68040 and the 486.

The fastest (20 MHz) 386 runs at 5 MIPS, the fastest (25 MHz) 68020 at 4 MIPS. Which is faster? Well, the 386 arrived 14 months later than the 68020. Keeping in mind that speeds are doubling every 24 months, the 386 should be a tad more than 1.5 times faster than the 68020—and it isn't. Besides, the 68030—the latest and fastest 680x0—is now real.

Do you see that a newly "announced" CPU chip (paper tiger) that has lots more performance than what you can buy today is no big deal? Even if it becomes real some day? Remember, the Zilog Z80000 was a world-devouring paper tiger in June 1983. Today it's real, but not very fast.

CPU performance is increasing by a factor of 32 every ten years. Memory bus-bandwidth (the product of bus width and memory speed) isn't increasing nearly that fast. For the future I suggest you place your money on CPU architectures that best use the scarcest computer resource: memory bus-bandwidth. And that means CISC.

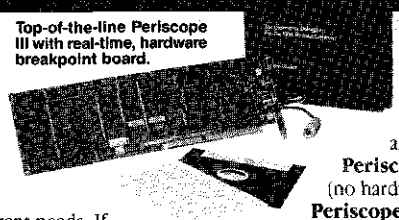
It would be very nice if that CPU also ran 1-2-3. Or Hypercards.

Afterword:
I have tried very hard to restrict this writeup to *current* production. All of the chip manufacturers have announced faster chips, smaller geometries, new features, etc. Many of these improvements will eventually become real because the industry is improving CPU performance at a predictable doubling per two years. But before you write to complain that, for instance, the Acorn ARM does too have a multiply instruction, you might check and see if the version they're shipping *now* has such an instruction. The Sun SPARC and the Mips chip are currently being shipped only by Sun and Mips and so are for the time being proprietary.

Today you can buy a 4 MIPS personal computer for less than the price of all but the cheapest compact cars. By the end of this century, you will be able to buy a 250 MIPS personal computer for about the same price in constant dollars. Which is a good thing, considering the ridiculously inefficient operating systems, software, and user interfaces that are being foisted on us . . . ♦

PERISCOPE™ • POWER

...Keeps you going full steam ahead when other debuggers let you down. With four models to pick from, you'll find a Periscope that has just the power you need.



Top-of-the-line Periscope III with real-time, hardware breakpoint board.

. . . Start with the model that fits your current needs. If you need more horsepower, upgrade for the difference in price plus $10! And don't worry about having a lot more to learn . . . Even when you move to the most powerful model, Periscope III, an extra dozen commands are all that's involved.

Periscope's software is solid, comprehensive, and flexible. It helps you debug just about any kind of program you can write . . . thoroughly and efficiently. Periscope's hardware adds the power to solve the really tough debugging problems.

Periscope requires an IBM PC, XT, AT, or close compatible (Periscope III requires hardware as well as software compatibility); DOS 2.0 or later; 64K available memory; one disk drive; an 80-column monitor.

Periscope I includes a half-length board with 56K of write-protected RAM; break-out switch; software and manual for $345.

Periscope II includes break-out switch; software and manual for $175.

Periscope II-X includes software and manual (no hardware) for $145.

Periscope III includes a full-length board with 64K of write-protected RAM, hardware breakpoints and real-time trace buffer; break-out switch; software and manual. Periscope III for machines running up to 8 MHz is $995; for machines running up to 10 MHz, $1095.

Call Toll-Free for free information or to order your Periscope today!

MAJOR CREDIT CARDS ACCEPTED. **800-722-7006**

The PERISCOPE Company, Inc.
1197 PEACHTREE ST.
PLAZA LEVEL
ATLANTA, GA 30361
404/875-8080

Number 6 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Quit Wasting Time!

As a programmer, most of your time is spent writing and debugging source code, and documenting your work. A powerful, easy-to-use programmable text editor could be saving you HOURS of unnecessary effort.

Only MULTI-EDIT has all these time-saving features:

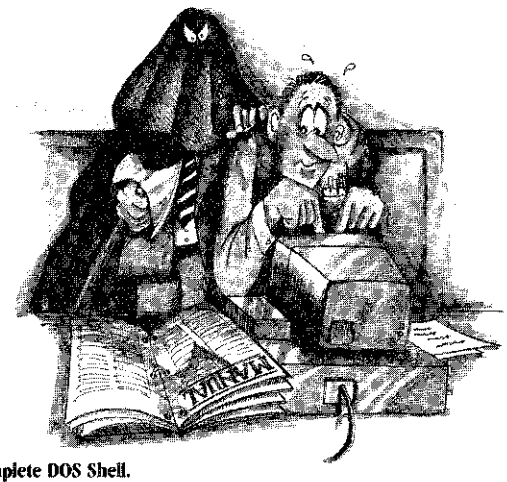
Fully automatic Windowing and Virtual Memory.
Edit multiple files regardless of physical memory size.
Easy cut-and-paste between files.
View different parts of the same file.

Powerful, EASY-TO-READ high-level macro language.
Standard language syntax.
Full access to ALL Editor functions.
Automate repetitive tasks.
Easy, automatic recording of keystrokes.

Language-specific macros for ALL major languages.
Smart indenting.
Smart brace/parenthesis/block checking.
Template editing.
Supports C, Pascal, BASIC and Assembler.

Terrific word-processing features for all your documentation needs.
Intelligent word-wrap.
Automatic pagination.
Full print formatting with justification, bold type, underlining and centering.
Even a table of contents generator.

Compile within the editor.
Automatically positions cursor at errors.
Built-in MAKE capabilities.
Run compiled program without leaving editor.
Automatically allocates all available memory to compiler or program.



Complete DOS Shell.
Scrollable directory listing.
Copy, Delete and Load multiple files with one command.
Background file printing.

Regular expression search and translate.
Condensed Mode display, for easy viewing of your program structure.
Pop-up FULL-FUNCTION Programmer's Calculator and ASCII chart.

and MOST IMPORTANT,
the BEST user-interface on the market!

- Extensive context-sensitive help.
- Choice of full menu system or logical function key layout.
- Function keys are always labeled on screen (no guessing required!).
- Excellent online, interactive tutorial.
- Keyboard may be easily reconfigured and re-labeled.

Users of Wordstar and Turbo Pascal's Editor could be programming in a fraction of the time with these features.

NO EDITOR ON THE MARKET TODAY HAS ALL THESE FEATURES, OR OFFERS YOU THIS MUCH POWER AT A REASONABLE PRICE, EXCEPT

Multi-Edit VERSION 2.0 $99. COMPLETE

Or Get our FULLY FUNCTIONAL DEMO Copy for only $10!

	Multi-Edit	BRIEF 2.0	Norton Editor	Vedit Plus
Edit 20+ files larger than memory	Yes	Yes	No	Yes
Powerful high level macro language	Yes	Yes	No	Yes
Full UNDO	Yes	Yes	No	No
Visual marking of blocks	Yes	Yes	Yes	No
Column oriented block operations	Yes	Yes	No	No
Automatic file save	Yes	Yes	No	No
Online help	Extensive	Limited	Limited	Limited
Online tutorial	Yes	No	No	Yes
Choice of keystroke commands or menu system	Yes	No	No	Yes
Function key assignments labeled on screen	Yes	No	No	No
WP Functions	Extensive	Limited	Limited	Extra Cost
Complete DOS shell	Yes	No	No	No
Pop-up Programmer's Calculator and ASCII table	Yes	No	No	No ASCII
Unlimited 'Off the Cuff' keystroke macros	Yes	No	No	Yes
Allocates all available memory to compiler when run from within editor	Yes	No	Yes	Yes
Intelligent indenting, template editing and brace/parenthesis/block matching and checking for all major languages	Yes	C Only	No	Limited
Flexible condensed mode display	Yes	No	Yes	No
Optional background communications and Spell Checker modules	Yes	No	No	No
PRICE	**$99**	**$195**	**$50**	**$185**

Requires IBM/PC/XT/AT/PS2 or full compatible, 256K RAM, PC/MS-DOS 2.0 or later-Multi-Edit and American Cybernetics are trademarks of American Cybernetics. BRIEF is a trademark of Underware, Inc. Norton Editor is a trademark of Peter Norton Computing, Inc. Vedit is a registered trademark of CompuView Products Inc. Copyright 1987 by American Cybernetics.

Get our FULLY FUNCTIONAL DEMO Copy for only $10!

To Order, Call 24 hours a day:
1-800-221-9280 Ext. 951
In Arizona: 1-602-890-1166
Credit Card and COD orders accepted

American Cybernetics
138 Madrid Plaza
Mesa, AZ 85201

Number 184 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Pink Noise

by
Frank D. Greco

Making Waves with the Raves

Let's face it, computer types are just as trendy as other people. I came up with the following list of "what's neat" and "what's beat" from my perspective. This is not a comparison of products or technologies—just a list of what's currently popular or unpopular. Because of the transient nature of trends, by the time you read this column, the list will probably be out-of-date.

What's Hot	What's Not
Hypermedia/Hypertext	Full-Screen Menu Selection
C++	ANSI C
Neural Networks	"Artificial Intelligence"
Object-Oriented Systems	Procedure-Based Systems
High-Resolution Imaging Software	Character Graphics
Macintosh II	IBM AT with Windows
Learning Object Oriented Programming	Learning C Language Programming
PostScript	Printer Escape Sequences
Sun Microsystems' Products	AT&T Computer Products
TI 34010	Intel 82786
Fiberoptic Ethernet	StarLAN
Apple Computer	IBM
NeWS and X Windows	MS Windows
Smalltalk	Prolog
DSPs	CISC Processors
Ray Duncan	Jean Yates and Jim Seymour
Sun's SPARC Chip	Motorola's 68030 Microprocessor
Superconductors and GaAS on Si	Ordinary Silicon Chips
Intel's P9 Chip	Intel's 486 Chip
Sybase and Oracle	dBase and Informix
Overlapping Windows	Tiled Windows
Fractal Data Compression	CCITT Group III/IV Compression
FDDI	Coax Ethernet
SQL	FRED and HAL
Holographic Scanning	Manual Digitization
Look and Feel Litigation	Interface Innovation
Steve Jobs	Steve Jobs
OS/2	OS/2
Programmer's Journal	*BYTE* Magazine

Comments, Rumor, and Conjecture

PARC Your (Hyper)Card, Sir

Although Apple's Bill Atkinson receives all the accolades for designing Big Red's first hypermedia-like application Hypercard, the congratulations for this achievement should also go (again) to Xerox. Most people don't know that Hypercard derives much of its essence from Notecards, an application developed by Xerox PARC researchers Frank Halasz, Thomas Moran, and Randall Trigg. Notecards is yet another Xerox idea that Apple has refined and capitalized on. While I think Bill Atkinson has another programming tour de force under his belt with Hypercard, much credit should go to the Xerox Notecards group and for that matter to the entire nascent hypertext/hypermedia culture (Englebart, Nelson, *et. al.*).

Really Ringing Her Bell

According to several colleagues inside Bell Laboratories, a deadly Captain Midnite is attacking Ma Bell's computer sites through AT&T's Datakit network and Usenet. The security group is baffled by this insidious computer villain, who leaves a massive wake of corrupt filesystems and deleted files. The phone company is so worried they've assigned their top computer security expert to the case.

P.S., It's Free

The University of Tokyo has placed a scaled-down version of the PostScript language in the public domain. I just received my copy of the source code from my Japanese contacts and am trying to find time to port it to all my machines. Apparently, it was originally written for the Toshiba line of PCs. From a quick once-over of the source (written in Microsoft C), it looks like it supports about 75 to 80 percent of "Red Book PostScript." It certainly covers the graphics portion of the language (the easy part to implement) but looks a little skimpy on font generation (the hard part). By the way, "Red Book PostScript" refers to the definition of the PostScript language in Adobe's *PostScript Language Reference Manual*, which is red. There will be more news of this item in the next Redirections.

No More FAT

In addition, I have it from close sources that the File Allocation Table (a.k.a. the "FAT") is slated for obsolescence. It will disappear in a future version of OS/2. There are signs that the new design borrows some file system technology from the Berkeley 4.2/4.3 Unix boys.

Borland for Dollars

In recent Turbo C advertisements, an interesting piece of C code is shown on the screen of the PC. The code appears to be performing some sort of process context switching. Task switching? Hmm . . . remember that Borland's Philippe Kahn has an obvious disdain for the 386 protected mode and an equal dislike for Microsoft's OS/2. Wouldn't it be interesting if Borland came out with a multitasking mechanism that uses pre-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

emptive lightweight processes in real mode? Similar products are out there, but none has the backing of a large software name like "Borland." By beating Microsoft to the punch, this product could be a real money-maker that would extend the life of MS-DOS.

As I've said in this column before, most multitasking applications don't need the significant memory protection the 286 and 386 offer in protected mode (i.e., OS/2). This blue-sky product would use the 286 in real mode with 32 megabytes of EMS 4.0 extended memory or the 386 in real mode with 32-bit addressing (one very large segment). Another interesting advantage offered by this scheme is instant backward compatibility with all other MS-DOS programs—the processor is still in *real* mode! Nifty, ain't it? Most applications don't need the complexity of OS/2—just the additional memory, that can be obtained through EMS 4.0 and some lightweight tasking mechanisms. If Borland can get their Codeview-competitive debugger to the market soon, along with a product like this, it might make for serious competition against the Microsoft boys and OS/2. Of course, Microsoft could just drop MS-DOS completely and force everyone to go with OS/2. However improbable that seems, it makes one think.

Speaking of Microsoft, I hear that some highly talented software types are thinking of leaving the company because it has grown too large and too corporate. Apparently, it was a cozy, stimulating place to work when it was a smaller company. Some of my California colleagues say similar things about Apple. Sun Microsystems, on the other hand, is attracting more software talent than any other computer company in the valley.

Could Mr. NeXT Be Mr. RiGHT?

Why has Steve Jobs been regularly calling a certain New Jersey-based Unix company? I hear it has absolutely nothing to do with bits and bytes, just birds and bees . . . ahem.

Amazing Stories

There's a grass roots movement to support HAM-style telecommunication. Someone has apparently managed to send data over the air at 56 Kbaud. From what this person told me, the TCP/IP standard has already been implemented over the physical layer using this amazing speed. A recent demonstration pulled a file off a Sun 3/280 on an Ethernet LAN in California onto a PC in New Jersey—all at 56,000 baud! The only catch to this technology is a legal one. The airwaves can't be used for commercial purposes without a license. I guess using the air for business transmissions makes you the equivalent of a radio station. Imagine performing remote procedure calls in a distributed processing environment completely wireless! Of course, it wouldn't be as noise-free as an optical fiber connection, but think of the many business applications. Because of the legal restrictions, this technology will probably be relegated to HAM/computer hackers sending files among friends to avoid Ma Bell's telephone charges.

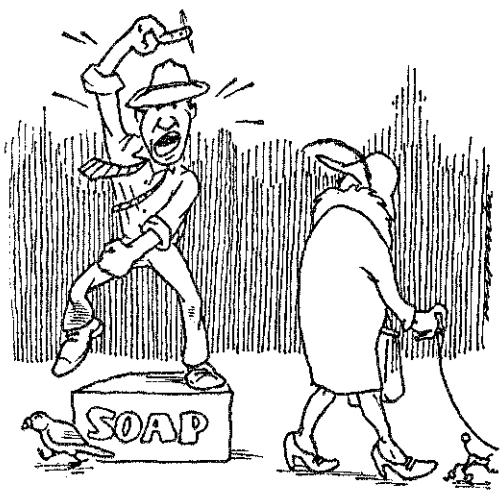
I Font to Be Alone

Now that PostScript is becoming *the* standard page description language (and Encapsulated PostScript, EPS, is trying to become the standard document description language) for printers and screens, systems programmers are using PostScript code as their graphics library. The Amazing Steve Jobs and His Fabulous NeXT Machine will put Adobe's new Display PostScript product into a ROM much in the way he put Bill Atkinson's hand-tuned QuickDraw routines into the Macintosh ROM. Because most of the industry press has confused this news item, let's clarify a few things. Adobe Systems developed Display PostScript. NeXT will use Display PostScript to develop an extensible and license-able window system based on the Adobe product. In other words, the NeXT window system (with Adobe's Display PostScript embedded) will compete with Sun's NeWS.

It will be only a matter of time before PostScript finds its way into the BIOS of all PC machines. Even the industry-popular but techno-damaged X Windows will use Display PostScript (although in a very rudimentary way). And with people like Phoenix, Control-C, and Conographics getting into the PS clone business, it looks like the postfix PostScript will be with us for a long while. By the way, check out Conographics' implementation. While I could not really check its supposed conformance with red book PostScript, the Conographics product is fast.

Wild Pitch

At PC Expo in New York City, AT&T announced their 3B4000 machine. They are targeting this machine at the transaction-based, fault-tolerant market. According to their press release, the 3B4000 uses several of their 32200 processors (ho-hum) in parallel fashion. The phone company claims that this machine will perform up to 43 MIPS and handle hundreds of concurrent users! Come on, Mama Bell. You expect



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

people to believe that after you told them that the original AT&T 3B2300 would handle 64 concurrent users when it could barely manage a handful of users?! If you took 100 AT-class machines and put them in a large enough box, you would have a computing engine capable of supplying at least 100 MIPS of power, and at substantially less cost than the 3B4000's. Everyone knows that marketing stretches the truth, but AT&T Marketing appears to be doing research in truth tensile-strength thresholds.

Positively POSIX

The effort that the U.S. government, the /usr/group organization, and the IEEE have put into the POSIX standard will strengthen the Unix operating system interface and perhaps weaken the market positioning of AT&T, especially in federal systems. Once the POSIX standard is solidified, companies such as DEC and Microsoft will be able to slightly modify, enhance, or cloak (i.e., POSIX-ify) their respective multitasking operating systems (VMS and OS/2) and bid on the very large jobs (I mean $uper large) for which AT&T's System V Unix previously had no challenger.[See Miscellaneous in this issue—Ed.] Why do you think the workstation wonder, Sun Microsystems, wants to merge AT&T's System V with their SunOS? It isn't because Sun thinks the marketing marvels at AT&T will help sell Sun workstations. Sun just wants those important large customer contracts.

It appears that Unix, or at least the Unix OS interface, will survive despite the efforts of AT&T.

Oh Yeah, Your Mother Boots Army Software

Saying that the Unix filesystem is breakable or that the Unix shell is not "user friendly" is like saying that MS-DOS has no installable device drivers and redirection, or that DEBUG can't understand natural language. When you hear someone say these things about Unix, you can bet good money they read *Datamation* or *ComputerWorld* (. . . those mainframe-heads . . .).

Sun Struck

Watching the Sun

Someone called recently to ask why I frequently mention Sun Microsystems in a PC oriented journal. First of all, I consider the Sun machines "PCs." They truly are personal computers, especially at today's prices (see below). Secondly, in an era when good software differentiates machines and prevents hardware from falling into the commodity market, Sun has been a technological leader. Any intelligent programmer, consultant, or computer scientist must keep an eye on the leaders. And with the tremendous price/performance of their current machines (Sun 3/60 and Sun 4 families), new VAR program, and strong industry alliances, I believe Sun has the potential to be the next DEC or IBM.

The Joy of Sun

Sun Vice President Bill Joy, architect of Berkeley Unix, csh, and the vi editor, has increasingly made his presence known in the industry. Within the last month, I must have read his name in the trade press about a dozen times. Apparently, Sun has decided that the newly clean-shaven Berkeley Bill's quotes make for good PR, hence "the tour." Joy was recently on the East Coast discussing Sun's future. I had the opportunity to listen to one of his interesting talks.

Joy's Ploys

Not surprisingly, the co-founder of Sun started with a standard marketing script—the new Sun slogan, "The Network is the Computer." It was amusing to see one of the true gurus of the Unix world try to be a computer salesman. At times, he appeared almost embarrassed to be hawking workstations to a highly technical audience. After this clumsy start, Joy discussed Sun's Network-extensible Window System (a.k.a. NeWS) and other windowing systems. He put down the current Sun windowing system: "Sunview is very, very old technology." I'd add, "Sunview is very, very cumbersome technology." Anyone who has programmed for the Sun workstation using Sunview knows what truly inspires a great wailing and gnashing of teeth—any of you MS Windows programmers out there know that feeling too?

There's No Joy in Mudville . . . He Was in Redmond, Washington

Comparing MIT's X Windows to NeWS, Joy said, "X Windows is also old technology. It is raster op versus NeWS's imaging technology." In other words, X Windows (MS Windows and Mac Windows, too) merely shuttles bits around in memory. Imaging goes past video bit-mapping and adds another sophisticated layer. Using graphic abstractions like the PostScript model, in effect, turns a bit-mapped screen into a PostScript screen. The video is no longer thought of as bit-mapped; it's PostScript-mapped. Another remark about the two window systems (note that I did not say "window managers"): "Think of X.11 as being the EBCDIC and NeWS as the ASCII."

According to Joy, Sun and Microsoft have an interesting joint agreement. Microsoft has a source license for NeWS and gets some sophisticated windowing technology from Sun. From Microsoft, Sun licensed—well, Joy said he couldn't say what it was or he would get into nondisclosure trouble. He did say, however, that it was very significant. Could it be that Sun is getting a NeWS emulation of Microsoft's Presentation Manager (a.k.a. MS Windows)? Or is it that Microsoft wants a networkable version of a PostScript-based MS Windows? Or is it both?

"cat" Got Your Tongue, Bill?

According to Joy, interfaces important to Sun are IBM's SAA, Microsoft's OS/2, Apple's Macintosh, and soft coprocessor technology (i.e., intelligent hardware emulation) that will allow MS-DOS to be run without additional hardware in a NeWS window on a Sun workstation. "MS-DOS then takes its correct place in the scheme of things—as a utility program like the [Unix] cat program." Despite this last tongue-in-cheek remark, Joy said that Sun expects a multi-user version of OS/2 to premier one year after the single-user OS/2 is delivered to end-users: "Because of basic economics, OS/2 and Unix will become the clear choices for operating systems in the future."

The last portion of Joy's talk revolved around the future of workstation technology in hardware and software. Joy said that object-oriented systems are here to stay and that languages such as Smalltalk and C++ are the languages of choice. In fact, a current research project in the Sun labs is a complete rewrite of the SunOS (Unix) kernel in CC++ (that's Concurrent C++ for those of you who don't know). True functional (as opposed to procedural) languages such as Prolog are not making the grade ("Prolog is not work-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Software Tools

❑ **The DATE UTILITIES** let you copy, erase, or display files based on their name **and** date. You can also change a file's time stamp. All utilities support wildcards for file names and let you limit the files chosen to those created **before**, **on**, or **after** a specified date. The package includes DCOPY, DERASE, DDIR, CDATE, and three other general purpose utilities. Version 1.5 costs $47.50.

❑ **PROINDEX** is a professional document indexing package. It works with all word processed text to create an index in any style. PROINDEX compares your text file to a **list** of keywords or phrases. These keywords **or** phrases can be up to 70 characters long! No limit on the size of the file to index or the number of references. Version 2.24 costs $89.95.

❑ **Soft Fonts** for the HP LaserJet and compatible printers. We offer **inexpensive** Helv, Roman, and Script font families. We have Roman fixed width, Bar code 39, Title fonts, Symbol sets, and Credit cards. All font families are $50 or less! Printer drivers for most word processors cost only $15 per font family! Custom fonts & a download utility are also available.

All tools work on the IBM PC or any compatible, MS-DOS 2.1 & up. No charge for shipping in the U.S. Thirty day money back guaranty!

ECI P.O. Box 61, Wasco, Illinois 60183

VISA 312-377-3520 MC

Number 70 on the Reader Service Card

FREE Demo Disk

I use Show Me! all day long. "...it's great!"

Show Me!, the pop-up multi-window file viewer, allows you to browse, print, search, and paste from any file. New *Version III* adds EMS and LAN support, multi-file text search, and more! Our customers swear by *Show Me!* You will too!

Escape... through the windows of *Show Me!*

Call for your FREE demo disk
(800) 634-3122

Single user version $59.95; unlimited user network version $119.95. Add $5 shipping & handling. Visa & Mastercard accepted. Unconditional 90 day satisfaction guarantee.

Serengeti Software • P.O. Box 27254 • Austin, Texas 78755

Number 161 on the Reader Service Card

The Pointer

The Memory Resident Help System and Demo Program Generator

The Pointer is the ultimate tool for creating powerful demo programs and creating help systems, all without programming! Here are just some of the features that are too numerous to list.

Help Features:

- Save hundreds of hours creating help systems for your programs
- Retro-fit old applications with a new help system in just minutes
- All commands are menu-driven for fast execution and easy learning
- Simply move the cursor to the word you want help for and press enter. A window will pop-up instantly with your help information
- Supports mice for fast cursor movement and menu selection
- No programming involved
- No source code generated for you to compile and link into your program
- No limit to the size of help system generated
- Runtime version requires only 68K

Demo Features:

- No limit to the size of your demo
- Create demos that:
  - are self-running
  - require user interaction
  - interact with and drive a live application program
  - combine all of the above
- Hand holding feature lets you create demos that will drive a live application, not just a picture show like those other demo programs
- Demos can be run from DOS or from inside your program
- Supports mice for effortless creation of windows and pointers
- Windows can be any size, up to a full screen
- Ideal for creating tutorials, demos and for prototyping

Best of all, you can create as many help systems and demos as you like. NO ROYALTIES EVER! Money Back guarantee.

ORDER NOW! Only $129.95 U.S.
(416)-270-6673
Stau Enterprises Inc.
3650 Kaneff Cres., Suite 1808, Mississauga Ont. Canada., L5A 4A1

STAU ENTERPRISES INC.

Number 134 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ing out"—Joy). I have always felt that the change in programming philosophy one must make before becoming very proficient with functional languages such as Prolog may not be worth it (the lack of speed using ordinary machine architectures is another issue).

SPARC's Flying

Sun's new Scaleable Processor ARChitecture (SPARC) 55,000 transistor chip is currently rated at 10 (VAX-style) MIPS. It's licensed to and manufactured by Fujitsu (hmm . . . don't they own 49.5% of Amdahl?). Joy pointed out that SPARC licensee Cypress Technologies (yet another Ben Rosen creation) will announce a 20 MIPS CMOS version by next summer. About another year after that, another SPARC licensee, Bipolar Integrated Technologies (BIT), should announce a 50+ MIPS version using ECL technology. That's not all—by 1990 a GaAS version of the chip will run at 100 MIPS. And according to a renowned GaAS expert Sun has just hired, advanced (and hybrid) GaAS technologies will allow the SPARC chip to run at 300 to 400 MIPS by 1992—all this without a parallel processing architecture! We haven't mentioned superconductor implementations yet! Of course, all of this is vaporware now, but who knows? Every year that Sun has been in business, they have promised to (at least) double CPU performance. So far, they haven't disappointed their clients.

That's Logical

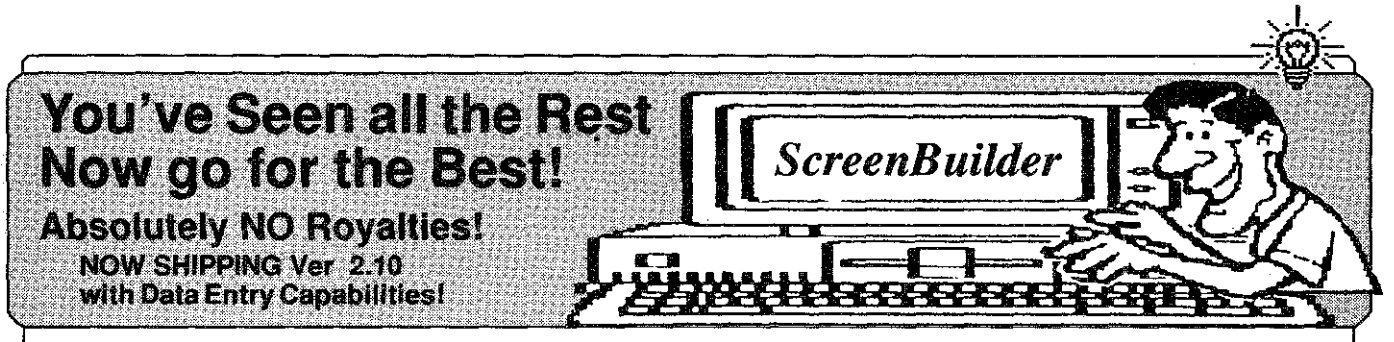
Another interesting feature of Sun's RISC-based SPARC design involves hardware emulation. According to some people in the know, Sun is researching the possibility of emulating Intel's 80386 (well, at least the opcodes). It's a fairly straightforward task to mimic the logical operations of a CISC using the small instruction set of a RISC. If Sun succeeds in copying the logical operations of a 386 (which suffice for most applications), you won't need a 386 to run OS/2 or MS-DOS. And, you won't have to depend on Intel to speed up the 386 clock to get incremental performance past 3–4 MIPS—use Sun's 50 MIPS SPARC in 1990. Now that's performance. Of course, Intel's lawyers might not like this too much.

Blinded Them with Science

High powered graphics and imaging technology are becoming more and more important in workstation applications. Joy said, for example, that just as you can change the shade or color of an object by, say, pulling on a slide bar on a Macintosh, an experiment of the future could be performed in which a chemist could view the effects of temperature on a molecule by simply moving a slide bar and seeing the effect. "We must get to the point where you're not animating graphics; you're animating science." Thought provoking. Joy closed his talk by theorizing about the turn-of-the-century Sun 9. Here are a few of his projections:

- 10 picosecond cycle time
- 100,000 MIPS
- 128 gigabytes—64 bit word architecture
- 4000 x 6000 (resolution) x 24 (frames/sec) x 2 (stereo) x 24 bits (colors)
- 30,000 megabits/sec transfer rates

After the talk, there were rumors that the price of the popular 2 MIPS Sun 3/50 68020-based diskless workstation



• **Create Interactive, Animated, Automated:**
- Demos
- Tutorials
- User/Program Shells
- Teaching Aids
- Children's Education
- Continuously Running Presentations

• **Orders of Magnitude above the rest:**
- Can generate BASIC BLOADable Screens
- Can generate Turbo C Source for Screens
- Can generate ANSI files.
- Sound and Music Capabilities
- Mathematics and String Logic
- User Inputs and Outputs
- Printer Control
- Loops, Subroutines, Decision Making
- Screens are directly usable by Application Programs
- Supports all 256 Colors and Characters
- Screen Capture Program
- Runs on CGA, EGA and Monochrome
- Much, Much More!

• **Easy to use. Has context sensitive help.**
- Complete, comprehensive 150+ page User Manual guides you through every step of the way.

Just $59.95 Complete!

Sample Demo Disk available for $2.00 (Credited towards your first order)

Call Today to order by credit card or send $59.95 plus $3.00 shipping/handling to:

GILMORE SYSTEMS
P. O. Box 3831
Beverly Hills, CA 90212-0831
(213) 275-8006

Requires IBM PC, XT, AT or compatible. DOS 2.0 or higher. 320 K RAM minimum. Fixed disk recommended.

Number 135 on the Reader Service Card

PRODUCED 2007 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

will be lowered to $3,000 in a few months. That includes 4 megabytes of RAM, a 16MHz 68020 processor with the 68881 floating point chip, 19-inch monochrome monitor with 1024 x 1024 resolution, Ethernet transceivers, full-blown SunOS (complete Berkeley and System V Unixes in one), keyboard, mouse, and a PostScript-based window system. Of course, this price would not include disk drives, but for those of you who already have local area networks and a fileserver, the price (if the rumor is true) is unbelievable. In addition, I overheard a Sun salesman say that Businessland is a serious contender for mass distribution of the 3/50 and the upcoming 386 workstation (which Daisy Systems already has). Add to this the recently announced VAR program, a port of SunOS to the NAS mainframe, and rumored IBM acceptance of NFS (IBM has already licensed Sun's XDR and RPC networking mechanisms), and Sun is setting themselves up for the big time. I wouldn't doubt that in the near future, Microsoft and Borland will be writing application software for the Sun machines. Sun represents an attractive untapped end-user market for both software vendors.

AI—Annoying Intelligence

[Soapbox ON] A common mistake when dialing a long distance phone number is to forget the initial "1" before the area code. What results is a recorded, "We're sorry. You must first dial a '1'. . . etc." Well, if the phone company has the intelligence to know that I forgot the "1," can't they insert the "1" and redial the darn number? Or how about inadvertently dialing the "1" when dialing a local number? You get, "It is not necessary to dial a '1' when calling this number . . . etc." Well, remove the "1" and please redial the darn number! I thought Ma Bell had better technology than that! Sorry, I had to get this one off my chest. [Soapbox OFF]

Redirection Reconsiderations

While reading past copies of Redirections (I kinda like my own column . . . so sue me), I noted that I said you cannot redirect standard error in Unix using the Berkeley C-Shell. What I should have said is that you cannot redirect standard error *cleanly* using the C-Shell. The redirection syntax of the Bourne and Korn shells is much more elegant.

To redirect **stderr** using **csh**, you must do the following:

```
% (pgm > stdout_file)  >& stderr_file
```

To redirect **stderr** using **sh** (Bourne) or **ksh** (Korn), it's "cleaner" to

```
%  pgm 2>stderr_file
```

All my identifiers in the examples above (**pgm**, **stdout_file**, and **stderr_file**) are completely arbitrary. I hope my earlier explanations didn't confuse anyone.

Thought of the Month

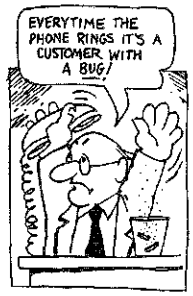
Will the word "hyper" soon become as ubiquitous and trendy as the word "turbo"? ♦

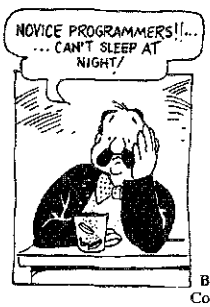
Frank D. Greco is an extremely busy independent consultant who specializes in efficient and effective computer systems. Readers can contact Frank at the following address:

Frank D. Greco
c/o Crossroads Technologies, Inc.
P.O. Box 530
Fanwood, New Jersey 07023
(201)-754-7820

Got Software Problems?

Get Turbo GhostWriter





Now $99.00

	Full Version $289	Starter Version $99	Cost of Upgrade for $99 version
B-Tree File Manager	Yes	Yes	N/A
Context-Sensitive Help	Yes		50
Command Window	Yes		50
Relational Model	Yes		150
Free Support	No Limit	1 Hour	100
Manual	Cloth D-ring	Paperback	50
All Upgrades	...	...	225

Turbo GhostWriter is an application generator that creates 80% of your custom application automatically -- as a Turbo Pascal Program. That **80%** is done swiftly in just 10% of the time it takes to code it yourself.That **80%** contains all of the "hooks" to add custom features such as table verification, security, currency conversion, importing files from other languages, etc...

Now it's up to you to write the code to relate one file to another. The relational model shows you how and it only takes a few minutes. End to end, all programming for 5 relational files should take less than 30 minutes. It's far superior to doing the whole thing from scratch. Your user sees a consistent interface, all programs function the same way, and your code is 100% error-free Turbo Pascal.

ASCII wants to give you a chance to try Turbo GhostWriter. For a *limited time you can buy* a fully functional version for just $99. This version will create flat files such as mailing lists, telephone files, inventory files, simple payroll files, etc. The utilities are the same as for the full-blown product, but are modestly packaged, and support is limited. It's the perfect way to see if an application generator is for you without risking a cent.

Orders & Information 800 227-7681
30 day money-back guarantee*(less $14 s/h)*

YES! Send me___copies of Turbo GhostWriter today.
Name _____
Address _____
City,State,Zip _____
Phone _____
MC, Visa, or Choice Card
#(or write COD) _____
Expiration Date _____
Circle version $99 $289
ASCII - 3239 Mill Run - Raleigh, NC 27612
800 227-7681

Number 170 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Yet Another VGA Write Mode

by
Michael Abrash

Installment 6: In which we venture ever deeper into unknown waters, encountering yet another of those damnable write modes.

In the last installment of *On Graphics*, we learned about the markedly peculiar write mode 3 of the VGA—after having spent four installments learning the ins and outs of the EGA's write mode 0, touching on write mode 1 as well. Happily, it turns out that the VGA supports all the write modes (write modes 0, 1, and 2) of the EGA and also supports the same read modes as the EGA. Which leaves two burning questions: What is write mode 2, and how the heck do you *read* VGA (and EGA) memory?

Write mode 2 is a bit unusual but not really hard to understand, particularly for those of you who followed my description of set/reset in *PJ*, Volume 5.4. Reading VGA memory, on the other hand, can be stranger than you would ever imagine. Let's start with the easy stuff, write mode 2. If there's room, I'll start in on reading VGA memory, but I suspect that will have to wait until next time.

Write Mode 2

Remember how set/reset worked? Good, because that's pretty much how write mode 2 works. (You *don't* remember? Well, I'll provide a brief refresher, but I suggest that you get *PJ* back issues 5.1 through 5.4 and 6.1 and come up to speed on the EGA and VGA.)

Recall that the set/reset circuitry for each of the four planes affects the byte written by the CPU in one of three ways: by replacing the CPU byte with 0, by replacing it with 0FFh, or by leaving it unchanged. The nature of the transformation for each plane is controlled by two bits. The enable set/reset bit for a given plane selects whether the CPU byte is replaced or not, and the set/reset bit for that plane selects the value with which the CPU byte is replaced if the enable set/reset bit is 1. The net effect of set/reset is to independently force any, none, or all planes to either all ones or all zeros on CPU writes. As we discussed a few articles back, this is a convenient way to force a specific color to appear no matter what color the pixels being overwritten are. Set/reset also allows the CPU to control the contents of some planes while the set/reset circuitry controls the contents of other planes.

Write mode 2 is basically a set/reset-type mode with enable set/reset always on for all planes and the set/reset data coming directly from the byte written by the CPU. Put another way, the lower four bits written by the CPU are written across the four planes, thereby becoming a color value. Put yet another way, bit 0 of the CPU byte is expanded to a byte and sent to the plane 0 ALU (if bit 0 is 0, a 0 byte is the CPU-side input to the plane 0 ALU, while if bit 0 is 1, a 0FFh byte is the CPU-side input); likewise, bit 1 of the CPU byte is expanded to a byte for plane 1, bit 2 is expanded for plane 2, and bit 3 is expanded for plane 3.

Those of you who have been following this series probably understand write mode 2 at this point; I suspect that the rest of you could use some additional explanation of an admittedly non-obvious mode. Let's follow the CPU byte through write mode 2, step by step.

Figure 1 shows the write mode 2 data path. The CPU byte comes into the VGA (or EGA) and is split into four separate bits, one for each plane. Bits 7–4 of the CPU byte vanish into the bit bucket, never to be heard from again. Speculation long held that those 4 unused bits indicated that IBM would someday come out with an 8-plane EGA that supported 256 colors. When IBM did finally come out with a 256-color mode (mode 13h of the VGA), it turned out not to be planar at all, and the upper nibble of the CPU byte remains unused in write mode 2 to this day.

The bit of the CPU byte sent to each plane is expanded to a 0 or 0FFh byte, depending on whether the bit is 0 or 1, respectively. The byte for each plane then becomes the CPU-side input to the respective plane's ALU. From this point on, the write mode 2 data path is identical to the write mode 0 data path. As discussed in earlier articles, the latch byte for each plane is the other ALU input, and the ALU either ANDs, ORs, or XORs the two bytes together or simply passes the CPU-side byte through. The byte generated by each plane's ALU then goes through the bit mask circuitry, which selects on a bit-by-bit basis between the ALU byte and the latch byte.

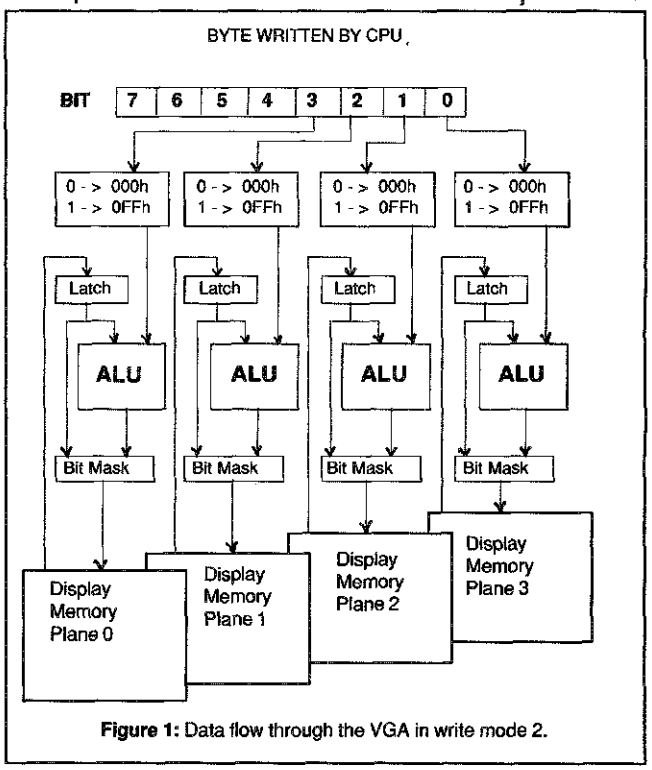


Figure 1: Data flow through the VGA in write mode 2.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Finally, the byte from the bit mask circuitry for each plane is written to that plane if the corresponding bit in the Map Mask register is set to 1.

It's worth noting the two differences between write mode 2 and write mode 0, the standard write mode of the VGA. First, rotation of the CPU data byte does not take place in write mode 2. Second, the Set/Reset and Enable Set/Reset registers have no effect in write mode 2.

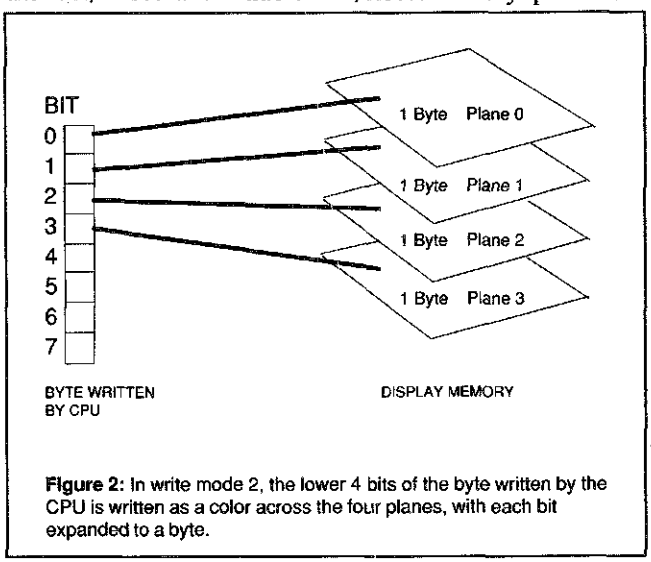


Figure 2: In write mode 2, the lower 4 bits of the byte written by the CPU is written as a color across the four planes, with each bit expanded to a byte.

Now that we understand the mechanics of write mode 2, we can step back and get a feel for what it might be useful for. View bits 3–0 of the CPU byte as a single pixel in one of sixteen colors. Next, imagine that nibble turned sideways and written across the four planes, one bit to a plane. Finally, expand each of the bits to a byte, as shown in Figure 2, so that 8 pixels are drawn in the color selected by bits 3–0 of the CPU byte. Within the constraints of the VGA's data paths, that's exactly what write mode 2 does.

By "the constraints of the VGA's data paths," I mean the ALUs, the bit mask, and the map mask. As Figure 1 indicates, the ALUs can modify the color written by the CPU, the map mask can prevent the CPU from altering selected planes, and the bit mask can prevent the CPU from altering selected bits of the byte written to. (Actually, the bit mask simply substitutes latch bits for ALU bits, but since the latches are normally loaded from the destination display memory byte, the net effect of the bit mask is usually to preserve bits of the destination byte.) These are not really constraints at all, of course, but rather features of the VGA; I simply want to make it clear that the use of write mode 2 to set 8 pixels to a given color is a rather simple special case among the many possible ways in which write mode 2 can be used.

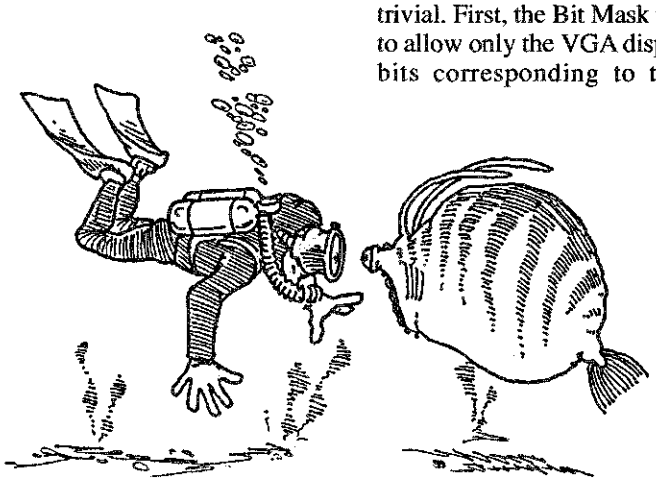
Write mode 2 is selected by setting bits 1 and 0 of the Graphics Mode register (indexed Graphics Controller register 5) to 1 and 0, respectively. Since VGA registers are readable, the correct way to select write mode 2 on the VGA is to read the Graphics Mode register, mask off bits 1 and 0, OR in 00000010b (03h), and write the result back to the Graphics Mode register, thereby leaving the other bits in the register undisturbed. Unfortunately, EGA registers are emphatically *not* readable, making it difficult to take advantage of the VGA's readable registers without separate EGA and VGA drivers.

Copying Chunky Bit-maps to VGA Memory Using Write Mode 2

Let's take a look at two examples of write mode 2 in action. Listing 1 shows a program that uses write mode 2 to copy a graphics image in chunky format to the VGA. In chunky format, adjacent bits in a single byte make up each pixel: Mode 4 of the CGA, EGA, and VGA is a 2 bit per pixel chunky mode, and mode 13h of the VGA is an 8 bit per pixel chunky mode. Chunky format is convenient since all the information about each pixel is contained in a single byte; consequently, chunky format is often used to store bit-maps in system memory.

Unfortunately, VGA memory is organized as a planar rather than chunky bit-map in modes 0Dh through 12h, with the bits that make up each pixel spread across four planes. The conversion from chunky to planar format in write mode 0 is quite a nuisance, requiring a good deal of bit manipulation. In write mode 2, however, the conversion becomes a snap, as shown in Listing 1. Once the VGA is placed in write mode 2, the lower four bits (the lower nibble) of the CPU byte (a single 4-bit chunky pixel) become eight planar pixels, all the same color. As discussed in *PJ*, 5.3, the bit mask makes it possible to narrow the effect of the CPU write down to a single pixel.

Given the above, conversion of a chunky 4-bit per pixel bit-map to the VGA's planar format in write mode 2 is trivial. First, the Bit Mask register is set to allow only the VGA display memory bits corresponding to the leftmost



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

chunky pixel of the two stored in the first chunky bit-map byte to be modified. Next, the destination byte in display memory is read in order to load the latches. Then a byte containing two chunky pixels is read from the chunky bit-map in system memory, and the byte is rotated four bits to the right to get the leftmost chunky pixel in position. This rotated byte is written to the destination byte; since write mode 2 is active, each bit of the chunky pixel goes to its respective plane, and since the Bit Mask register is set up to allow only one bit in each plane to be modified, a single pixel in the color of the chunky pixel is written to VGA memory.

The above process is then repeated for the right-most chunky pixel, if necessary, and repeated again for as many pixels as there are in the image.

"That's an interesting application of write mode 2," you may well say, "but is it really useful?" While the ability to convert chunky bit-maps into VGA bit-maps does have its uses, Listing 1 is primarily intended to illustrate the mechanics of write mode 2. In point of fact, I've heard of a considerably faster (and truly ingenious) way to write chunky pixel bit-maps to EGA memory using write mode 2, read mode 1, and the graphics position registers; sadly, since the VGA lacks graphics position registers, the faster conversion technique works only on the EGA, so it is no longer really useful.

Drawing Color-patterned Lines Using Write Mode 2

A more serviceable use of write mode 2 is shown in Listing 2. The program shown in Listing 2 draws multicolored horizontal, vertical, and diagonal lines, basing the color patterns on passed color tables. Write mode 2 is ideal because in this application color can vary from one pixel to the next, and in write mode 2 all that's required to set pixel color is a change of the lower nibble of the byte written by the CPU. Set/reset could be used to achieve the same result, but an index/data pair of OUTs would be required to set the Set/Reset register to each new color. Similarly, the Map Mask register could be used in write mode 0 to set pixel color, but in this case not only would an index/data pair of OUTs be required, but there would also be no guarantee that data already in display memory wouldn't interfere with the color of the pixel being drawn since the Map Mask register allows only selected planes to be drawn to.

Listing 2 is hardly a comprehensive line drawing program since it draws only a few special line cases, and although it is reasonably fast, it is far from the fastest possible code to handle those cases since it goes through a dot-plot routine and since it draws horizontal lines a pixel rather than a byte at a time. Write mode 2 would, however, serve just as well in a full-blown line drawing routine. For any type of patterned line drawing on the VGA, or indeed for any type of patterned drawing at all, the basic principles remain the same: Use the Bit Mask to select the pixel (or pixels) to be altered, and use the CPU byte in write mode 2 to select the color in which the pixel is to be drawn.

When to Use Write Mode 2 and When to Use Set/reset

As indicated above, write mode 2 and set/reset are functionally interchangeable. Write mode 2 lends itself to more efficient implementations when the drawing color changes frequently, as in Listing 2.

Set/reset tends to be superior when many pixels in succession are drawn in the same color since with set/reset enabled for all planes, the Set/Reset register provides the color data and as a result, the CPU is free to draw whatever byte value it wishes. For example, the CPU can execute an OR instruction to display memory when set/reset is enabled for all planes, thus both loading the latches and writing the color value with a single instruction, secure in the knowledge that the value it writes is ignored in favor of the set/reset color.

Set/reset is also the mode of choice whenever it is necessary to force the value written to some planes to a fixed value while allowing the CPU byte to modify other planes. This is the mode of operation when set/reset is enabled for some but not all planes.

Mode 13h—320 x 200 with 256 Colors

I've been asked several times recently about the programming model for mode 13h, the VGA's 320 x 200 256-color mode. Frankly, there's just not much to it. Mode 13h offers the simplest programming model in the history of PC graphics: a linear bit-map at A000:0000 consisting of 64,000 bytes, each controlling one pixel. The byte at offset 0

Programmers and Information Specialists,

Simplify Your Life.

Learn 3270 HLLAPI

In our intensive ONE DAY SEMINAR you will learn to develop intelligent micro-mainframe links using the PC or PS/2™ and the IBM® industry standard High Level Language Application Program Interface (HLLAPI). Instruction includes both fundamentals and specific programming techniques.

Seminar TOPICS:

- Introduction to Application Program Interface (API)
- Application Ideas and Development
- Learning the Major HLLAPI Services

Participants receive Attachmate's API PROGRAMMER'S TOOLKIT, normally $195, featuring full documentation and programming examples of Attachmate's IBM-compatible HLLAPI.

Seminar fee is $295 per person

Attachmate

Quality Micro-Mainframe Solutions

REGISTER NOW! CALL 1-800-426-6283

Attachmate Corp., 3241 118th S.E. Bellevue, WA 98005, 206-644-4010

IBM® is a registered trademark and PS/2 is a trademark of International Business Machine Corporation.

Number 178 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

controls the upper left pixel on the screen, the byte at offset 13Fh controls the upper right pixel on the screen, the byte at offset 140h controls the second pixel down at the left of the screen, and the byte at offset 63,999 controls the lower right pixel on the screen. That's all there is to it; it's so simple that I'm not going to waste your time with a demo program. If you do desire a demo program for mode 13h (and modes 11h and 12h as well), a forthcoming issue of the Borland magazine *Turbo Technix* (possibly Volume 1, Number 4, but that's currently up in the air) will provide just that in the form of "The VGA Standard" by Yours Truly.

Flipping Pages from Text to Graphics and Back

In the past week or so I've gotten both a phone call and a letter on an interesting EGA/VGA topic; I didn't remember the name of the caller, but the question is unusual enough that both probably came from the same person. At any rate, the letter came from Phil Coleman, of La Jolla, who writes,

> Suppose I have the EGA in mode $10 (color 640 x 350 graphics). I would like to preserve some or all of the image while I temporarily switch to text mode 3 to give my user a "Help" screen. Naturally, memory is scarce, so I'd rather not make a copy of the video buffer at $A000 to "remember"the image while I digress to the Help text. The EGA BIOS says that the screen memory will not be cleared on a mode set if bit 7 of AL is set. Yet if I try that, it is clear that writing text into the $B800 buffer trashes much more than the 4 Kbytes of a text page; when I switch back to mode $10, "ghosts" appear in the form of bands of colored dots. (When in text mode, I do make a copy of the 4K buffer at $B800 before showing the help; and I restore the 4K before switching back to mode $10.) Is there a way to preserve the graphics image (or at least >24K of it) while I switch to text mode?
>
> A corollary to this question: Where does the 64/128/256K of EGA memory "hide" when the EGA is in text mode? Some, I guess, is used to store character sets, but what happens to the rest? Or rather, how can I protect it?

Those are good questions. Alas, answering them in full would require considerable explanation, and while I intend to do that some day, now is not the time. However, the issue of how to go to text mode and back without losing the graphics image is worth discussing briefly.

Phil is indeed correct in his observation that setting bit 7 of AL instructs the BIOS not to clear display memory on mode sets, and he is also correct in surmising that a font is loaded when going to text mode. The normal mode 10h bit-map occupies the first 28,000 bytes of each of the VGA's four planes. The normal mode 3 character/attribute memory map resides in the first 4000 bytes of planes 0 and 1 (the blue and green planes in mode 10h). The standard font in mode 3 is stored in the first 8K of plane 2 (the red plane in mode 10h). Neither mode 3 nor any other text mode makes use of plane 3; if necessary, plane 3 could be used as scratch memory in text mode.

Consequently, you can certainly get away with saving a total of just under 16 Kbytes—the first 4000 bytes of planes 0 and 1 and the first 8 Kbytes of plane 2—when going from mode 10h to mode 3, to be restored on returning to mode 10h.

That's hardly all there is to the matter of going from text to graphics and back without bit-map corruption, though. One interesting point is that the mode 10h bit-map can be relocated to A000:8000 simply by doing a mode set to mode 10h and setting the start address (programmed at CRT Controller registers 0Ch and 0Dh) to 8000h. You can then access display memory starting at A800:0000 instead of the normal A000:0000, with the resultant display exactly like that of normal mode 10h. (There are BIOS issues since the BIOS doesn't automatically access display memory at the new start address, but if your program does all its drawing directly without the help of the BIOS, that's no problem.)

At any rate, once the mode 10h bit-map is relocated to A800:0000, flipping to text mode and back becomes painless. The memory used by mode 3 doesn't overlap the relocated mode 10h bit-map at all, so all you need do is set bit 7 of AL on mode sets.

Another interesting point about flipping from graphics to text and back is that the standard mode 3 character/attribute map doesn't actually take up every byte of the first 4000 bytes of planes 0 and 1. The standard mode 3 charac-

NEW PRODUCTS! NEW PRODUCTS! NEW PRODUCTS!

ANNOUNCING THE MOST POWERFUL DATABASE GRAPHICS DEVELOPMENT AND ANALYSIS SYSTEM AVAILABLE

G-WIZ®

- **Full featured interactive presentation graphics development system.**
- *Works with all popular printers, plotters, cameras and slidemakers.*
- Smart scaling routines for ten datatypes provide the fastest route to perfect graphs.
- Fully supports all popular screen devices including: CGA, EGA, VGA, DGIS, and Hercules monochrome.
- Uses maximum output device resolution.
- **Introductory Price, $395.00**

G-RUN™

- **Has all the output features of G-WIZ above plus:**
- Eliminates the need for end-users to learn a graphics package.
- Produces full-featured presentation graphics from within dBASE, R:base, and many other programming languages.
- Thanks to the incredibly robust x-axis and y-axis scaling/labeling routines, this is the ONLY integrated database graphics system available that can produce perfect graphs from live data without operator intervention.
- **Introductory Price, $195.00**

dPROBE®

- **dBASE II, III, and III+, Foxbase, and Clipper source code documentation and analysis.**
- *More than 13 different reports on program/variable structure available.*
- Time-saving table of contents organized in two different ways!
- An invaluable aid to consultants and application developers! Completely automates program documentation and structure.
- **Introductory Price, $89.00**

Offered to you by MIGHTYSOFT Inc., P.O. Box 51048, Seattle, Washington 98115. To order call: (206) 525-3766. Telex: 6503329229. Visa and Mastercharge welcome.

Number 179 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ter/attribute map actually takes up only, every even byte of the first 4000 in each plane; the odd bytes are left untouched. This means that only about 12 Kbytes actually have to be saved when going to text mode. The code in Listing 3 flips from graphics mode to text mode and back, saving only those 12 Kbytes that actually have to be saved. This code saves and restores the first 8K of plane 2 (the font area) while in graphics mode, but saves and restores the 4000 bytes used for the character/attribute map in text mode since the characters and attributes, which are actually every other byte of planes 0 and 1, respectively, appear to be contiguous bytes in memory in text mode and so are easily saved as a single block.

Explaining why only every other byte of planes 0 and 1 is used in text mode and why characters and attributes appear to be contiguous bytes when they are actually in different planes is a large part of the explanation I haven't room to go into now. One bit of fallout from this, however, is that if you flip to text mode and preserve the graphics bit-map using the mechanism illustrated in Listing 3, you shouldn't write to any text page other than page 0 (that is, don't write to any offset in display memory above 3999 in text mode) or alter the Page Select bit in the Miscellaneous Output register (3C2h) while in text mode. In order to allow unfettered access to text pages, it would be necessary to save the first 32K of each of planes 0 and 1. (On the other hand, this would allow up to 16 text screens to be stored simultaneously, with any one displayable instantly—a topic for yet another installment.) Moreover, if any fonts other than the default font are loaded, the portion of plane 2 those particular fonts are loaded into would have to be saved, up to a maximum of all 64K of plane 2. In the worst case, a full 128K would have to be saved in order to preserve all the memory potentially used by text mode.

As I said, Phil Coleman's question is a complex one, and I've only touched on the intriguing possibilities arising from the various configurations of display memory in VGA graphics and text modes. Someday I'll return to it, but for now we've still got the basics of the remarkably complex VGA to cover.

Until Next Time

As I suspected, we didn't get around to learning how to read VGA memory. That's probably just as well since that leaves the whole next installment in which to cover that fairly esoteric subject. If there are any other EGA or VGA topics you'd like to see covered, write to me in care of *PJ* or at the post office box listed below, or drop me a line on MCIMail (user name: MABRASH), and I'll try to get to it soon. See you next time. ♦

Michael Abrash is an Engineering Fellow currently working on advanced graphics projects for Video Seven, Inc., in Fremont, California. Readers may write to Michael at

P.O. Box 390351
Mountain View, CA 94039

Code follows.

If you see something here you like, by all means, compare it with what our competition has to offer.

Seidl Version Manager (SVM) Version 2.0 is a comprehensive source code configuration system designed for professional software developers. SVM tracks changes made to individual modules (revisions) and stores "version" information that tells which revisions go into making which versions of a product. SVM supports multiple developers in both networking and non-networking environments. Some of its features include: revision branching, revision merging, user ids and optional passwords, seven different file access levels, a menu driven shell, formatted reports with user definable layouts, central journal logging, on-line help, optional data compression, a remote site update utility, and more. ***$299.95, LAN version $1000.00***

Seidl Make Utility (SMK) Version 2.1 is the most sophisticated product generation utility available. SMK is not just another copy of the of the UNIX make. SMK uses a structured, high-level, *dependency definition language (DDL)* that is a programming language in itself. It supports symbolic constants, parameterized macros, for loops, if-then-else style conditionals, interactive statements, ambiguous path name expressions, and much more. In addition, SMK is lightening fast and won't bog down, even on your largest projects. ***$99.95, LAN version $500.00 (includes SMKgen).***

New!* SMKgen Version 1.0** is an automatic dependency generation utility. By examining the files that make up a product, SMKgen determines how those files depend on each other and amazingly writes a dependency file that can be processed by SMK. SMKgen can also construct link files and *filelist* files that integrate with SVM. With SMKgen, the task of placing a software project under SMK is reduced to executing a single command. ***$49.95

"SVM is a full-featured system that has all the essential capabilities you are likely to need, and is further distinguished by several important features that none of its competitors yet have." ***- The C Users Journal™, February, 1988.***

When you know the facts, you'll know we're the best.

For more information call: **1-313-662-8086**
For a free copy of our competitive comparison report, write to us on your company letterhead. MC/VISA/COD accepted.

SEIDL COMPUTER ENGINEERING

3106 Hilltop Dr., Ann Arbor, MI 48103

Number 174 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

THE DOS USER'S GROUP

SUBSCRIBE NOW

Annual Membership
$25.00 (U.S.)
$35.00 (Foreign)

Quarterly Newsletter

- DOS Programming Hints
- DOS Command Overview & Tutorials
- DOS Customization Hints
- DOS Question & Answer Section
- OS/2 Commands & Concepts
- OS/2 Trends & Impacts
- OS/2 Question & Answer Section
- OS/2 Programming Hints
- MS Windows In Detail
- Third Party Product Reviews

Latest DOS and OS/2 Information

P.O Box 26601 • Las Vegas, NV 89126

Number **128** on the Reader Service Card

PLOT TEXT ON ANY GRAPHICS SCREEN!!

YES, we said ANY graphics screen, even VGA! ***Finally!*** **Xgraf** is a super set of low level assembly graphics routines that you call directly from Compiled Basic. The **Xgraf** kernel replaces Basic's confusing statements with consistent full featured calls specifically designed for the Basic programmer. **Xgraf** supports **Hercules, CGA, EGA, EEGA, VGA, and user defined screen sizes.** All modes can be accessed in real time - no recompiling!

Advanced features extend Basic graphic capabilities!

In addition to Basic's graphics features, **Xgraf's** ***advanced features*** include zooming, tiling, a virtual screen page, screen compression to disk, windowing, screen importing, *and much more!*

Finally! **Xgraf is only $99.00 + $4.00 s&h**
(Xgraf requires the Microsoft QuickBASIC compiler)

We *specialize* in libraries and tools for Compiled Basic. Our catalog features the ***Finally!*** family of programmer's libraries and *other top flight tools.*

KOMPUTERWERK - YOUR BASIC RESOURCE

KOMPUTERWERK
851 Parkview Blvd.
Pittsburgh, PA 15215

Call: 1-800-423-3400
PA & AK call (412) 782-0384
VISA and MC accepted!

Number **155** on the Reader Service Card

FORTRAN for the 80386 with source level symbolic debugging

- Use All Available Memory
- Run in the 32-bit Protected Mode of the 80386
- Support for 80287, 80387, and Weitek 1167
- Royalty Free License on Developed Programs

Put the full power of the 80386 into your programs. Our **SVS FORTRAN-386** compiler breaks DOS's 16-bit instruction and 640K memory limit. Cut a typical program's run time in half, while using up to 16 Megabytes of memory. **SVS FORTRAN-386** supports the INTEL 80287 and 80387 and the Weitek 1167 numerical co-processor chips for additional speed. We use IGC's powerful VM/RUN loader and execution manager, included with the compiler, to overcome the DOS limitation.

SVS FORTRAN-386 comes with a powerful **source level symbolic debugger.** The debugger lets the developer view the source codes, access and modify variables by symbolic name, set breakpoints and tracepoints, and reinitialize execution of the program. The debugger requires no modification to the object code, thus letting the user debug the production version of the program. In addition, the program runs under the debugger at full speed until a breakpoint is reached.

Features:

- Full implementation of the ANSI standard FORTRAN77 plus extensions.
- Powerful source level symbolic debugger.
- Comprehensive IEEE floating point software including single precision, double precision and complex arithmetic.
- High-speed compilation generating 32-bit instructions and utilizing all available memory for both code and data.
- Extensive code optimization with inline coding of many intrinsics, optimal register management, loop optimization and more.
- Mature product based on 6 years of development and use.

System Requirements: IBM PS/2 Model 80, COMPAQ Deskpro 386™ and Deskpro 386-20™, Intel Inboard 386™, or other computers based on the Intel motherboard
DOS 2.x or 3.x
INTEL 80287 or 80387, or Weitek 1167 numerical co-processors.

Price:

SVS	FORTRAN-386	$895
SVS	C-386	$895
SVS	Pascal-386	$895

Royalty-free license on developed programs.

Support: Free telephone technical support to registered users. Our support staff has extensive experience in porting and developing programs using SVS FORTRAN-386.

Available from:

SAIC®
Science Applications International Corporation
5150 El Camino Real Suite C-31
Los Altos, California 94022
(415) 960-5931

VM/RUN is a trademark of IGC. COMPAQ and DESKPRO 386 are registered trademark and trademark, respectively, of COMPAQ Computer Corporation. IBM is a registered trademark and Personal System/2 is a trademark of International Business Machines Corporation. Inboard 386 is a trademark of Intel Corporation.

Number **156** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

These listings are available on 360K, MS-DOS disk. See the ad on page 76 for ordering information.

Code From "Write Mode" by Michael Abrash — 1

```
;
; *** Listing 1 ***
;
; Program to illustrate one use of write mode 2 of the VGA and EGA by
; animating the image of an "A" drawn by copying it from a chunky
; bit-map in system memory to a planar bit-map in VGA or EGA memory.
;
; Assembled with MASM 5.0, linked with MS-LINK 3.60.
;
; By Michael Abrash 11/8/87
;
Stack   segment para stack 'STACK'
        db      512 dup(0)
Stack   ends

SCREEN_WIDTH_IN_BYTES   equ     80
DISPLAY_MEMORY_SEGMENT  equ     0a000h
SC_INDEX        equ     3c4h    ;Sequence Controller Index register
MAP_MASK        equ     2       ;index of Map Mask register
GC_INDEX        equ     03ceh   ;Graphics Controller Index reg
GRAPHICS_MODE   equ     5       ;index of Graphics Mode reg
BIT_MASK        equ     8       ;index of Bit Mask reg

Data    segment para common 'DATA'
;
; Current location of "A" as it is animated across the screen.
;
CurrentX        dw      ?
CurrentY        dw      ?
RemainingLength dw      ?
;
; Chunky bit-map image of a yellow "A" on a bright blue background
;
AImage          label   byte
                dw      13, 13          ;width, height in pixels
                db      000h, 000h, 000h, 000h, 000h, 000h, 000h
                db      009h, 099h, 099h, 099h, 099h, 099h, 000h
                db      009h, 099h, 099h, 099h, 099h, 099h, 000h
                db      009h, 099h, 099h, 0e9h, 099h, 099h, 000h
                db      009h, 099h, 09eh, 0eeh, 099h, 099h, 000h
                db      009h, 099h, 0eeh, 09eh, 0e9h, 099h, 000h
                db      009h, 09eh, 0e9h, 099h, 0eeh, 099h, 000h
                db      009h, 09eh, 0eeh, 0eeh, 0eeh, 099h, 000h
                db      009h, 09eh, 0e9h, 099h, 0eeh, 099h, 000h
                db      009h, 09eh, 0e9h, 099h, 0eeh, 099h, 000h
                db      009h, 099h, 099h, 099h, 099h, 099h, 000h
                db      009h, 099h, 099h, 099h, 099h, 099h, 000h
                db      000h, 000h, 000h, 000h, 000h, 000h, 000h
Data    ends

Code    segment para public 'CODE'
        assume  cs:Code, ds:Data
Start   proc    near
        mov     ax,Data
        mov     ds,ax
        mov     ax,10h
        int     10h             ;select video mode 10h (640x350)
;
; Prepare for animation.
;
        mov     [CurrentX],0
        mov     [CurrentY],200
        mov     [RemainingLength],600   ;move 600 times
;
; Animate, repeating RemainingLength times. It's unnecessary to erase
; the old image, since the one pixel of blank fringe around the image
; erases the part of the old image not overlapped by the new image.
;
AnimationLoop:
        mov     bx,[CurrentX]
        mov     cx,[CurrentY]
        mov     si,offset AImage
        call    DrawFromChunkyBitmap    ;draw the "A" image
        inc     [CurrentX]              ;move one pixel to the right
        dec     [RemainingLength]
        jnz     AnimationLoop
;
; Wait for a key before returning to text mode and ending.
;
        mov     ah,01h
        int     21h
        mov     ax,03h
        int     10h
        mov     ah,4ch
        int     21h
Start   endp
;
; Draw an image stored in a chunky-bit map into planar VGA/EGA memory
; at the specified location.
;
; Input:
;       BX = X screen location at which to draw the upper left corner
;               of the image
;       CX = Y screen location at which to draw the upper left corner
;               of the image
;       DS:SI = pointer to chunky image to draw, as follows:
;               word at 0: width of image, in pixels
;               word at 2: height of image, in pixels
;               byte at 4: msb/lsb = first & second chunky pixels,
;                       repeating for the remainder of the scan line
;                       of the image, then for all scan lines. Images
;                       with odd widths have an unused null nibble
;                       padding each scan line out to a byte width
;
; AX, BX, CX, DX, SI, DI, ES destroyed.
;
DrawFromChunkyBitmap    proc    near
        cld
;
; Select write mode 2.
;
        mov     dx,GC_INDEX
        mov     al,GRAPHICS_MODE
        out     dx,al
        inc     dx
        mov     al,02h
        out     dx,al
;
; Enable writes to all 4 planes.
```

2

```
        mov     dx,SC_INDEX
        mov     al,MAP_MASK
        out     dx,al
        inc     dx
        mov     al,0fh
        out     dx,al
;
; Point ES:DI to the display memory byte in which the first pixel
; of the image goes, with AH set up as the bit mask to access that
; pixel within the addressed byte.
;
        mov     ax,SCREEN_WIDTH_IN_BYTES
        mul     cx              ;offset of start of top scan line
        mov     di,ax
        mov     cl,bl
        and     cl,111b
        mov     ah,80h          ;set AH to the bit mask for the
        shr     ah,cl           ; initial pixel
        shr     bx,1
        shr     bx,1
        shr     bx,1            ;X in bytes
        add     di,bx           ;offset of upper left byte of image
        mov     bx,DISPLAY_MEMORY_SEGMENT
        mov     es,bx           ;ES:DI points to the byte at which the
                                ; upper left of the image goes
;
; Get the width and height of the image.
;
        mov     cx,[si]         ;get the width
        inc     si
        inc     si
        mov     bx,[si]         ;get the height
        inc     si
        inc     si
        mov     dx,GC_INDEX
        mov     al,BIT_MASK
        out     dx,al           ;leave the GC Index register pointing
        inc     dx              ; to the Bit Mask register
RowLoop:
        push    ax              ;preserve the left column's bit mask
        push    cx              ;preserve the width
        push    di              ;preserve the destination offset
ColumnLoop:
        mov     al,ah
        out     dx,al           ;set the bit mask to draw this pixel
        mov     al,es:[di]      ;load the latches
        mov     al,[si]         ;get the next two chunky pixels
        shr     al,1
        shr     al,1
        shr     al,1
        shr     al,1            ;move the first pixel into the lsb
        stosb                   ;draw the first pixel
        ror     ah,1            ;move mask to next pixel position
        jc      CheckMorePixels ;is next pixel in the adjacent byte?
        dec     di              ;no
CheckMorePixels:
        dec     cx              ;see if there are any more pixels
        jz      AdvanceToNextScanLine ; across in image
        mov     al,ah
        out     dx,al           ;set the bit mask to draw this pixel
        mov     al,es:[di]      ;load the latches
        lodsb                   ;get the same two chunky pixels again
                                ; and advance pointer to the next
                                ; two pixels
        stosb                   ;draw the second of the two pixels
        ror     ah,1            ;move mask to next pixel position
        jc      CheckMorePixels2 ;is next pixel in the adjacent byte?
        dec     di              ;no
CheckMorePixels2:
        loop    ColumnLoop      ;see if there are any more pixels
                                ; across in the image
        jmp     short CheckMoreScanLines
AdvanceToNextScanLine:
        inc     si              ;advance to the start of the next
                                ; scan line in the image
CheckMoreScanLines:
        pop     di              ;get back the destination offset
        pop     cx              ;get back the width
        pop     ax              ;get back the left column's bit mask
        add     di,SCREEN_WIDTH_IN_BYTES
                                ;point to the start of the next scan
                                ; line of the image
        dec     bx              ;see if there are any more scan lines
        jnz     RowLoop         ; in the image
        ret
DrawFromChunkyBitmap    endp
Code    ends
        end     Start
```

```
;
; *** Listing 2 ***
;
; Program to illustrate one use of write mode 2 of the VGA and EGA by
; drawing lines in color patterns.
;
; Assembled with MASM 5.0, linked with MS-LINK 3.60.
;
; By Michael Abrash 11/8/87
;
Stack   segment para stack 'STACK'
        db      512 dup(0)
Stack   ends

SCREEN_WIDTH_IN_BYTES   equ     80
GRAPHICS_SEGMENT equ    0a000h  ;mode 10 bit-map segment
SC_INDEX        equ     3c4h    ;Sequence Controller Index register
MAP_MASK        equ     2       ;index of Map Mask register
GC_INDEX        equ     03ceh   ;Graphics Controller Index reg
GRAPHICS_MODE   equ     5       ;index of Graphics Mode reg
BIT_MASK        equ     8       ;index of Bit Mask reg

Data    segment para common 'DATA'
Pattern0        db      16
                db      0, 1, 2, 3, 4, 5, 6, 7, 8
                db      9, 10, 11, 12, 13, 14, 15
Pattern1        db      6
                db      2, 2, 2, 10, 10, 10
Pattern2        db      8
                db      15, 15, 15, 0, 0, 15, 0, 0
Pattern3        db      9
                db      1, 1, 1, 2, 2, 2, 4, 4, 4
Data    ends
```


Code continues. . .

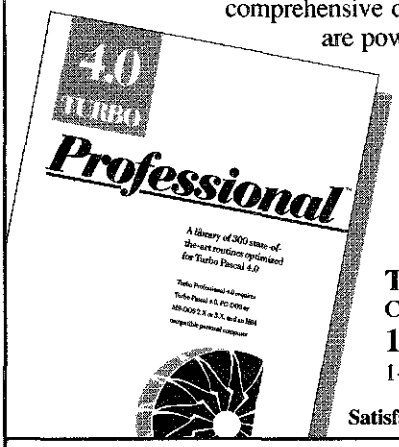
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

New!

Hire a Pro for Your New Turbo 4.0

Turn on the power of Turbo PROFESSIONAL 4.0, a library of more than 300 state-of-the-art routines optimized for Turbo Pascal 4.0. You'll have professional quality programs finished faster and easier.

Turbo PROFESSIONAL 4.0 includes complete source code, comprehensive documentation and demo programs that are powerful and useful. The routines include:



- Pop-up resident routines
- BCD arithmetic
- Virtual windows and menus
- EMS and extended memory access
- Long strings, large arrays, macros, and much more.

Turbo PROFESSIONAL is only $99.
Call toll free for credit card orders.
1-800-538-8157 extension 830
1-800-672-3470 extension 830 in CA

Satisfaction Guaranteed or your money back within 30 days.

Turbo Pascal 4.0 is required. Registered owners of Turbo Professional by Sunny Hill Software may upgrade for $30. Include your serial number.

For other information call 408-438-8608, 9 AM to 5 PM PST. Shipping & taxes prepaid for US and Canadian customers, others please add $6 per item.

TURBO Power

TurboPower Software 3109 Scotts Valley Dr., Suite 122 Scotts Valley, CA 95066

Number **25** on the Reader Service Card

Fast answers, $189.

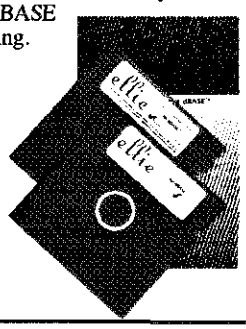
If you use dBASE, we'd like you to know about ELLIE. It's the first English query translator for the relational database standard.

We've *completed* the Information Circuit: IBM provided the hardware... Ashton-Tate the data access software. Until now, that left you with one large task when you needed the answer to a new question. Programming it.

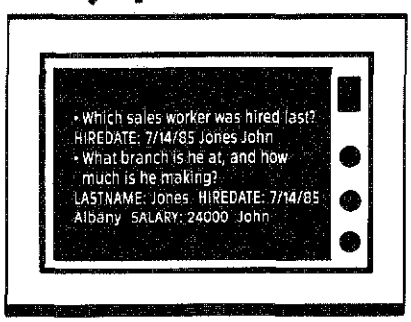
Today you can add the third component of the triad: ELLIE—the English Language Frontend. Now, to get answers from your data, all you do is ask.

Which of the sales workers that completed our advanced training course had higher third-quarter sales than Edwards?

Just ask. ELLIE translates your question into high-quality program (.prg) code, and at the touch of a key flashes control to dBASE for processing.



New! Version 1.1 features advanced memory-residency utilities.


• Which sales worker was hired last?
HIREDATE: 7/14/85 Jones John
• What branch is he at, and how much is he making?
LASTNAME: Jones HIREDATE: 7/14/85
Albany SALARY: 24000 John

dBASE, dBASE III are trademarks of Ashton-Tate Corp.

List the customers that ordered both widgets and wadgets since May. Which American League player with over 300 at bats had the highest on-base percentage? Show the current employer of applicants with BASIC experience, sorting by application date.

After nearly five years of development, how can we offer ELLIE for a mere $189? There are over a million dBASE users. None of them should be without it.

- Auto cross-references up to 5 relational files
- Self-customizing—reads your .dbf files direct
- On-the-fly unit conversion • 512K min. RAM
- For dBASE III/III Plus • (212) 316-9078

ELFSOFT
210 W 101 NY NY 10025

Number **113** on the Reader Service Card

TRUE MULTITASKING

With

MultiDos Plus

"multitasking for the IBM-PC."

Ideal for developing applications in process control, data acquisition, communications, and other areas. Check these features which make **MultiDos Plus** an unbeatable value.

- Run up to 32 programs concurrently.
- Your software continues to run under DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in most languages.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/resume programs.
- Programmatic interface via INT 15H for the following.
  - * Intertask message communication. Send/receive/check message present on 64 message queues.
  - * Task control by means of semaphores. Get/release/check semaphores.
  - * Change priority-256 priority levels.
  - * Suspend task for specified interval.
  - * Spawn and terminate external and internal tasks.
  - * Disable/enable multitasking.
  - * and more!
- Independent foreground/background displays.
- Access to DOS while applications are running.

Hardware/Software Requirements

IBM PC/XT/AT or true clone. Enough memory to hold **MultiDos Plus** (48 KB) and all your application programs. Also may need 4 or 16 KB memory for "hidden screens" for each active task. MS-DOS (or PC-DOS) 2.0 or later operating system.

only: $24.95 *OR* $99.95 with source code

Outside USA add $5.00 shipping and handling.
Visa and Mastercard orders only call toll-free: 1-800-872-4566, ext. 350., or send check or money order to:

NANOSOFT
13 Westfield Rd, Natick, MA 01760
MA orders add 5% sales tax.

Number **109** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
Code    segment para public 'CODE'
        assume  cs:Code, ds:Data
Start   proc    near
        mov     ax,Data
        mov     ds,ax
        mov     ax,10h
        int     10h             ;select video mode 10h (640x350)
;
; Draw 8 radial lines in upper left quadrant in pattern 0.
;
        mov     bx,0
        mov     cx,0
        mov     si,offset Pattern0
        call    QuadrantUp
;
; Draw 8 radial lines in upper right quadrant in pattern 1.
;
        mov     bx,320
        mov     cx,0
        mov     si,offset Pattern1
        call    QuadrantUp
;
; Draw 8 radial lines in lower left quadrant in pattern 2.
;
        mov     bx,0
        mov     cx,175
        mov     si,offset Pattern2
        call    QuadrantUp
;
; Draw 8 radial lines in lower right quadrant in pattern 3.
;
        mov     bx,320
        mov     cx,175
        mov     si,offset Pattern3
        call    QuadrantUp
;
; Wait for a key before returning to text mode and ending.
;
        mov     ah,01h
        int     21h
        mov     ax,03h
        int     10h
        mov     ah,4ch
        int     21h
;
; Draws 8 radial lines with specified pattern in specified mode 10h
; quadrant.
;
; Input:
;       BX = X coordinate of upper left corner of quadrant
;       CX = Y coordinate of upper left corner of quadrant
;       SI = pointer to pattern, in following form:
;               Byte 0: Length of pattern
;               Byte 1: Start of pattern, one color per byte
;
; AX, BX, CX, DX destroyed
;
QuadrantUp      proc    near
        add     bx,160
        add     cx,87           ;point to the center of the quadrant
        mov     ax,0
        mov     dx,160
        call    LineUp          ;draw horizontal line to right edge
        mov     ax,1
        mov     dx,88
        call    LineUp          ;draw diagonal line to upper right
        mov     ax,2
        mov     dx,88
        call    LineUp          ;draw vertical line to top edge
        mov     ax,3
        mov     dx,88
        call    LineUp          ;draw diagonal line to upper left
        mov     ax,4
        mov     dx,161
        call    LineUp          ;draw horizontal line to left edge
        mov     ax,5
        mov     dx,88
        call    LineUp          ;draw diagonal line to lower left
        mov     ax,6
        mov     dx,88
        call    LineUp          ;draw vertical line to bottom edge
        mov     ax,7
        mov     dx,88
        call    LineUp          ;draw diagonal line to bottom right
        ret
QuadrantUp      endp
;
; Draws a horizontal, vertical, or diagonal line (one of the eight
; possible radial lines) of the specified length from the specified
; starting point.
;
; Input:
;       AX = line direction, as follows:
;               3  2  1
;               4  *  0
;               5  6  7
;       BX = X coordinate of starting point
;       CX = Y coordinate of starting point
;       DX = length of line (number of pixels drawn)
;
; All registers preserved.
;
; Table of vectors to routines for each of the 8 possible lines.
;
LineUpVectors   label   word
        dw      LineUp0, LineUp1, LineUp2, LineUp3
        dw      LineUp4, LineUp5, LineUp6, LineUp7
;
; Macro to draw horizontal, vertical, or diagonal line.
;
; Input:
;       XParm = 1 to draw right, -1 to draw left, 0 to not move horz.
;       YParm = 1 to draw up, -1 to draw down, 0 to not move vert.
;       BX = X start location
;       CX = Y start location
;       DX = number of pixels to draw
;       DS:SI = line pattern
;
MLineUp macro   XParm, YParm
        local   LineUpLoop, CheckMoreLine
        mov     di,si           ;set aside start offset of pattern
        lodsb                   ;get length of pattern
        mov     ah,al
LineUpLoop:
        lodsb                   ;get color of this pixel...
        call    DotUpInColor    ;...and draw it
if XParm EQ 1
        inc     bx
```



```
endif
if XParm EQ -1
        dec     bx
endif
if YParm EQ 1
        inc     cx
endif
if YParm EQ -1
        dec     cx
endif
        dec     ah              ;at end of pattern?
        jnz     CheckMoreLine
        mov     si,di           ;get back start of pattern
        lodsb
        mov     ah,al           ;reset pattern count
CheckMoreLine:
        dec     dx
        jnz     LineUpLoop
        jmp     LineUpEnd
        endm

LineUp  proc    near
        push    ax
        push    bx
        push    cx
        push    dx
        push    si
        push    di
        push    es
        mov     di,ax
        mov     ax,GRAPHICS_SEGMENT
        mov     es,ax
        push    dx              ;save line length
;
; Enable writes to all planes.
;
        mov     dx,SC_INDEX
        mov     al,MAP_MASK
        out     dx,al
        inc     dx
        mov     al,0fh
        out     dx,al
;
; Select write mode 2.
;
        mov     dx,GC_INDEX
        mov     al,GRAPHICS_MODE
        out     dx,al
        inc     dx
        mov     al,02h
        out     dx,al
;
; Vector to proper routine.
;
        pop     dx              ;get back line length
        shl     di,1
        jmp     cs:[LineUpVectors+di]
;
; Horizontal line to right.
;
LineUp0:
        MLineUp 1, 0
;
; Diagonal line to upper right.
;
LineUp1:
        MLineUp 1, -1
;
; Vertical line to top.
;
LineUp2:
        MLineUp 0, -1
;
; Diagonal line to upper left.
;
LineUp3:
        MLineUp -1, -1
;
; Horizontal line to left.
;
LineUp4:
        MLineUp -1, 0
;
; Diagonal line to bottom left.
;
LineUp5:
        MLineUp -1, 1
;
; Vertical line to bottom.
;
LineUp6:
        MLineUp 0, 1
;
; Diagonal line to bottom right.
;
LineUp7:
        MLineUp 1, 1

LineUpEnd:
        pop     es
        pop     di
        pop     si
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        ret
LineUp  endp
;
; Draws a dot in the specified color at the specified location.
; Assumes that the VGA is in write mode 2 with writes to all planes
; enabled and that ES points to display memory.
;
; Input:
;       AL = dot color
;       BX = X coordinate of dot
;       CX = Y coordinate of dot
;       ES = display memory segment
;
; All registers preserved.
;
DotUpInColor    proc    near
        push    bx
        push    cx
        push    dx
        push    di
;
; Point ES:DI to the display memory byte in which the pixel goes, with
```

Code continues. . .

PRODUCED 2003 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LAHEY SETS NEW FORTRAN STANDARDS

LAHEY PERSONAL FORTRAN 77 - The full ANSI 77 Standard and Debugger for $95.00.

F77L - "EDITOR'S CHOICE" *PC Magazine*
"...the most robust compiler tested." *Micro/Systems Journal*
"...the ultimate fully-blown ANSI FORTRAN 77... a fantastic product." *PC Australia*

F77L-EM - Breaks the DOS 640K program barrier with extended memory. The most powerful PC FORTRAN available.

FORTRAN TOOLS: Profiler, Mathematical Functions Library, Overlay Linker, Utility Libraries, Widows Library, Toolkit.

The difficult decision is not from whom you should buy your PC FORTRAN, but which Lahey compiler you should purchase.

Call us today to discuss your PC FORTRAN needs.

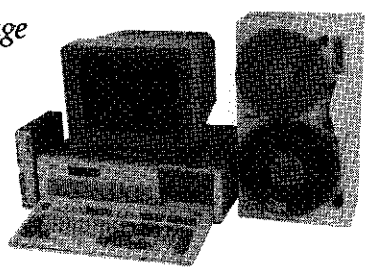
FOR INFORMATION OR TO ORDER:
1-800-548-4778
30-Day Money-Back Satisfaction Guarantee
Lahey Computer Systems, Inc., Box 6091, Incline Village, NV 89450 Tel: 702-831-2500 TLX: 9102401256

Lahey Computer Systems Inc.

Number 44 on the Reader Service Card

9-Track Tape Subsystem

The Data Interchange Solution You've Been Waiting For!



Qualstar's ½" 9-track Ministreamers™ bring full ANSI data interchange capability to your PC or Macintosh™ computer system.

With 9-track tape, you are free to exchange data files with any mainframe or minicomputer in the world.

Our affordable Ministreamers come in both 7" and 10½" versions and use less desk space than an ordinary sheet of paper. They can provide 1600 thru 6250 BPI capability and may be used for disk backup as well as data interchange. Complete subsystem prices, including software, start as low as $2,495 for 7" units and at $3,670 for 10½" units.

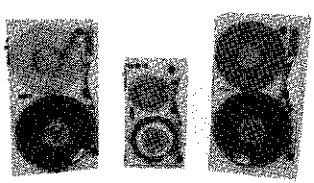
Qualstar has become the market leader in desk-top 9-track systems for a good reason; our tape drives have established an outstanding record for reliability and low cost of ownership.

Discover the many advantages 9-track tape has over other Micro/Mainframe links.

Call us today!

QUALSTAR®

9621 Irondale Avenue,
Chatsworth, CA 91311
Telephone: (818) 882-5822



Macintosh is a trademark of Apple Computer, Inc.

Number 97 on the Reader Service Card

Software Tools

For Programmers & Non-Programmers

Get 'State of the Art' performance and save valuable time with these high quality utilities!

Opt-Tech Sort™

Opt-Tech Sort is a high performance Sort/Merge/Select utility. It can read, sort and write a file faster than most programs can even read the data. Example: 1,000 records of 80 bytes can be read, sorted and a new file written in less than 10 seconds (IBM XT). Opt-Tech Sort can be used as a stand-alone program or called as a subroutine to over 25 different programming languages.

All the sorting, record selection and reformatting facilities you need are included. A partial list of features includes: The ability to process files of any size. Numerous filetypes are supported including Sequential, Random, Delimited, Btrieve, dBASE II & III and many others. Up to 10 key fields can be specified (ascending or descending order). Over 16 different types of data supported. Powerful record selection capability allows you to specify which records are to be included on your output. Record reformatting allows you to change the structure of your output record and to output special fields such as record numbers for use as indexes.

MS-DOS $149. ★ NEW ★ Xenix $249.

★ NEW ★ VERSION On-Line Help™ ★ NEW ★ VERSION

On-Line Help allows you to easily add "Help Windows" to all your programs. On-Line Help is actually two help packages in one. You get BOTH Resident (pop-up) and Callable Help Systems.

The resident version allows you to add help to any system. Your Help System is activated when the "Hot Keys" that you specify are pressed. You can then chain between help windows in any manner you desire.

The callable version allows you to easily display help windows from your programs. A simple call to the help system makes the window appear. The original screen is automatically restored when the help window is cleared. On-Line Help is callable from over 20 different languages.

You have full control over the help window content, size, color and location.

MS-DOS $149. **Demo $10.** (apply toward purchase).

Scroll & Recall™

Scroll & Recall is a resident screen and keyboard enhancement. It allows you to conveniently scroll back through data that has gone off the top of your display screen. Up to 27 screens of data can be recalled or written to a disk file (great for documenting systems operations). Also allows you to easily recall and edit your previously entered DOS commands without retyping.

Scroll & Recall is very easy to use. It's a resident utility that's always there when you need it. MS-DOS $69.

Visa, M/C, AMEX, Check, Money Order, COD or Purchase Orders accepted.

To order or to receive additional information just call and receive immediate highly qualified attention!

Opt-Tech Data Processing
P.O. Box 678 — Zephyr Cove, NV 89448
(702) 588-3737

Number 16 on the Reader Service Card

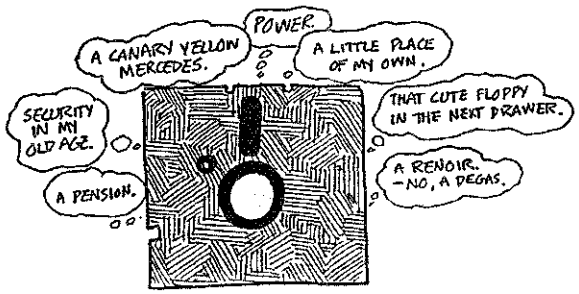
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
; the bit mask set up to access that pixel within the addressed byte.
;
        push    ax              ;preserve dot color
        mov     ax,SCREEN_WIDTH_IN_BYTES
        mul     cx              ;offset of start of top scan line
        mov     di,ax
        mov     cl,bl
        and     cl,111b
        mov     dx,GC_INDEX
        mov     al,BIT_MASK
        out     dx,al
        inc     dx
        mov     al,80h
        shr     al,cl
        out     dx,al           ;set the bit mask for the pixel
        shr     bx,1
        shr     bx,1
        shr     bx,1            ;X in bytes
        add     di,bx           ;offset of byte pixel is in
        mov     al,es:[di]      ;load latches
        pop     ax              ;get back dot color
        stosb                   ;write dot in desired color
        pop     di
        pop     dx
        pop     cx
        pop     bx
        ret
DotUpInColor    endp
Start   endp
Code    ends
        end     Start



;
; *** Listing 3 ***
;
; Program to illustrate flipping from bit-mapped graphics mode to
; text mode and back without losing any of the graphics bit-map.
;
; Assembled with MASM 5.0, linked with MS-LINK 3.60.
;
; By Michael Abrash 11/8/87
;
Stack   segment para stack 'STACK'
        db      512 dup(0)
Stack   ends

GRAPHICS_SEGMENT equ    0a000h  ;mode 10 bit-map segment
TEXT_SEGMENT    equ     0b800h  ;mode 3 bit-map segment
SC_INDEX        equ     3c4h    ;Sequence Controller Index register
MAP_MASK        equ     2       ;index of Map Mask register
GC_INDEX        equ     3ceh    ;Graphics Controller Index register
READ_MAP        equ     4       ;index of Read Map register

Data    segment para common 'DATA'

GStrikeAnyKeyMsg0       label   byte
        db      0dh, 0ah, 'Graphics mode', 0dh, 0ah
        db      'Strike any key to continue...', 0dh, 0ah, '$'

GStrikeAnyKeyMsg1       label   byte
        db      0dh, 0ah, 'Graphics mode again', 0dh, 0ah
        db      'Strike any key to continue...', 0dh, 0ah, '$'

TStrikeAnyKeyMsg        label   byte
        db      0dh, 0ah, 'Text mode', 0dh, 0ah
        db      'Strike any key to continue...', 0dh, 0ah, '$'

Plane2Save      db      2000h dup (?)   ;save area for plane 2 data
                                        ; where font gets loaded
CharAttSave     db      4000 dup (?)    ;save area for memory wiped
                                        ; out by character/attribute
                                        ; data in text mode
Data    ends

Code    segment para public 'CODE'
        assume  cs:Code, ds:Data
Start   proc    near
        mov     ax,10h
        int     10h             ;select video mode 10h (640x350)
;
; Fill the graphics bit-map with a colored pattern.
;
        cld
        mov     ax,GRAPHICS_SEGMENT
        mov     es,ax
        mov     ah,3            ;initial fill pattern
        mov     cx,4            ;four planes to fill
        mov     dx,SC_INDEX
        mov     al,MAP_MASK
        out     dx,al           ;leave the SC Index pointing to the
        inc     dx              ; Map Mask register
FillBitMap:
        mov     al,10h
        shr     al,cl           ;generate map mask for this plane
        out     dx,al           ;set map mask for this plane
        sub     di,di           ;start at offset 0
        mov     al,ah           ;get the fill pattern
        push    cx              ;preserve plane count
        mov     cx,8000h        ;fill 32K words
        rep stosw               ;do fill for this plane
        pop     cx              ;get back plane count
        shl     ah,1
        shl     ah,1
        loop    FillBitMap
;
; Put up "strike any key" message.
;
        mov     ax,Data
        mov     ds,ax
        mov     dx,offset GStrikeAnyKeyMsg0
        mov     ah,9
        int     21h
;
; Wait for a key.
;
        mov     ah,01h
        int     21h
;
; Save the 8K of plane 2 that will be used by the font.
;
        mov     dx,GC_INDEX
        mov     al,READ_MAP
        out     dx,al
        inc     dx
        mov     al,2
        out     dx,al           ;set up to read from plane 2
```



```
        mov     ax,Data
        mov     es,ax
        mov     ax,GRAPHICS_SEGMENT
        mov     ds,ax
        sub     si,si
        mov     di,offset Plane2Save
        mov     cx,2000h/2      ;save 8K (length of default font)
        rep movsw
;
; Go to text mode without clearing display memory.
;
        mov     ax,083h
        int     10h
;
; Save the text mode bit-map.
;
        mov     ax,Data
        mov     es,ax
        mov     ax,TEXT_SEGMENT
        mov     ds,ax
        sub     si,si
        mov     di,offset CharAttSave
        mov     cx,4000/2       ;length of one text screen in words
        rep movsw
;
; Fill the text mode screen with dots and put up "strike any key"
; message.
;
        mov     ax,TEXT_SEGMENT
        mov     es,ax
        sub     di,di
        mov     al,'.'          ;fill character
        mov     ah,7            ;fill attribute
        mov     cx,4000/2       ;length of one text screen in words
        rep stosw
        mov     ax,Data
        mov     ds,ax
        mov     dx,offset TStrikeAnyKeyMsg
        mov     ah,9
        int     21h
;
; Wait for a key.
;
        mov     ah,01h
        int     21h
;
; Restore the text mode screen to the state it was in on entering
; text mode.
;
        mov     ax,Data
        mov     ds,ax
        mov     ax,TEXT_SEGMENT
        mov     es,ax
        mov     si,offset CharAttSave
        sub     di,di
        mov     cx,4000/2       ;length of one text screen in words
        rep movsw
;
; Return to mode 10h without clearing display memory.
;
        mov     ax,90h
        int     10h
;
; Restore the portion of plane 2 that was wiped out by the font.
;
        mov     dx,SC_INDEX
        mov     al,MAP_MASK
        out     dx,al
        inc     dx
        mov     al,4
        out     dx,al           ;set up to write to plane 2
        mov     ax,Data
        mov     ds,ax
        mov     ax,GRAPHICS_SEGMENT
        mov     es,ax
        mov     si,offset Plane2Save
        sub     di,di
        mov     cx,2000h/2      ;restore 8K (length of default font)
        rep movsw
;
; Put up "strike any key" message.
;
        mov     ax,Data
        mov     ds,ax
        mov     dx,offset GStrikeAnyKeyMsg1
        mov     ah,9
        int     21h
;
; Wait for a key before returning to text mode and ending.
;
        mov     ah,01h
        int     21h
        mov     ax,03h
        int     10h
        mov     ah,4ch
        int     21h
Start   endp
Code    ends
        end     Start
```

♦

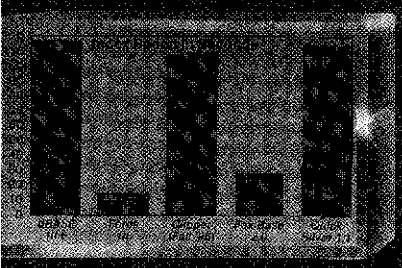

POWER.
A CANARY YELLOW MERCEDES.
A LITTLE PLACE OF MY OWN.
SECURITY IN MY OLD AGE.
THAT CUTE FLOPPY IN THE NEXT DRAWER.
A RENOIR. —NO, A DEGAS.
A PENSION.
-DISK DRIVES-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

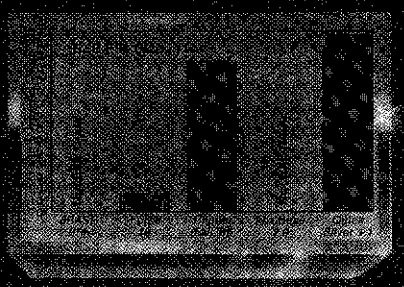
FORCE III

The Ultimate dBase Compiler

FORCE III is the world's fastest dBase compiler. It compiles and links the quickest, and the .EXE files it creates run so fast you'll swear you wrote your program in C. To give you an idea how fast, take a look at the *Index Speed* graph below. This graph compares indexing speeds of the different products using a database which contained 8,000 records. The database was indexed on the numeric field, NUM.



Best of all, FORCE III creates very small .EXE files. The .EXE files are *true* native code files and do not rely on any kind of memory resident library or interpreter! The reasons the .EXE files are so small are simple: FORCE III is an "intelligent" compiler and only uses those routines from the library that your program requires, and secondly, the entire library is written in Assembler, which makes the routines very small and very fast.



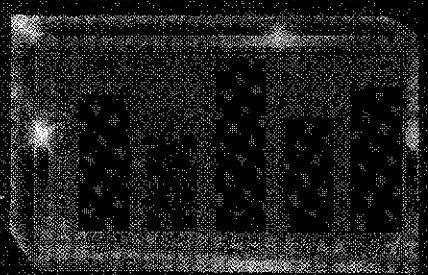
In the *Index Size* graph (below), the index files created by each product are compared. As you can see, not only is FORCE III the fastest at indexing, it also creates the smallest index files!

(Tests conducted on an IBM 8Mhz. AT with 20Mb hard disk.)

$129

Royalty Free and fully linkable!

In addition to being the fastest true dBase compiler on the market, FORCE III generated .EXE files carry no royalties whatsoever! And FORCE III compiled programs can be linked to Borland's Turbo C and other C compilers using the Intel/Microsoft OBJ format. And linking to Assembler is a snap. We even included an ASCII graphics generator for easy screen and menu generation.



Arrays, UDFs, sound, new functions, and more!

Force III includes features you've come to expect in more expensive dBase compilers. For the programmer we have included I/O re-direction, FOR-NEXT loops, C like file primitives, and new data types. We've even included some Clipper compatible commands such as VALID.

Call now, 800-922-3001.

FORCE III comes with extras, like a 120 day warranty, an excellent reference manual and for the first 10,000 customers, we'll include the premier sourcework, **Advance dBASE III Plus Programming and Techniques** by Miriam Liskin (a $21.95 value). All this for just $129.00 (add $5.00 s&h). In Colorado call 303-444-1542. In Europe call London at 44-1-631-0548. Or write: Sophco, Inc., PO Box 7430, Boulder, Colorado, 80306-7430.

Products mentioned above are trademarked by their respective manufacturers.

Number 136 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Inside Turbo Pascal 4.0

by
John J. Newlin

Tom Swan previewed an early Beta version of Turbo Pascal 4.0 in our November/December 1987, issue (Volume 5.6). Now John Newlin reviews the production compiler.

Four years ago Borland International was a small, one product company operated by a crazy Frenchman. Today, Borland International is a very large, multi-product company operated by a brilliant Frenchman. The difference is largely attributable to Borland's first product—Turbo Pascal. I well remember how amazed and delighted I was when I first tried Turbo Pascal. I had just completed a medium sized project, written in IBM Pascal, that required 25 minutes and four diskette changes to compile and link, so Turbo Pascal's quick, interactive editor/compiler environment was a mind-blowing experience.

Although Borland upgraded Turbo Pascal several times, the improvements were confined mainly to increased compile time performance and the addition of various tool boxes. Even though the compiler matured as a result of the upgrades, no substantive changes to the underlying architecture of the compiler relieved it of its original limitations. Third party support for Turbo Pascal permitted the creation of large, complex programs, but the compiler still suffered from the lack of code optimization and almost nonexistent project management facilities.

Now Borland has released Version 4.0 of Turbo Pascal, and it is much, much more than just an upgrade of Turbo 3.0. In fact, TP 4 is an entirely new compiler possessing features that make it a very attractive choice for serious software development projects. And although TP 4 has been redesigned from the ground up, it retains almost all the facilities that made its progenitor so successful.

The primary shortcoming of TP 3, in my estimation, was its inability to create and manage modules of separately compiled code. With TP 3, the programmer was forced to recompile the entire program any time a change to the source code had to be made. A related shortcoming of TP 3 was the fact that the entire runtime library was always present in the executable image. For example, a null program compiled under Turbo 3.01a occupies 11,433 bytes of code. The same program (nothing but a **begin end** statement pair) compiled with TP 4 occupies only 1360 bytes, which represents the essential startup and initialization code.

The original Turbo Pascal had a strong UCSD Pascal heritage, especially evident in its implementation of strings and string handling routines. TP 4 benefits even more from that heritage by implementing separately compiled units. A TP 4 unit is, for all intents and purposes, a complete subprogram that can contain all the declarations, constructs, and functionality of a main program. In fact, programmers will quickly learn that a well-organized, efficient TP 4 program will contain a very small main program supported by multiple separately compiled units.

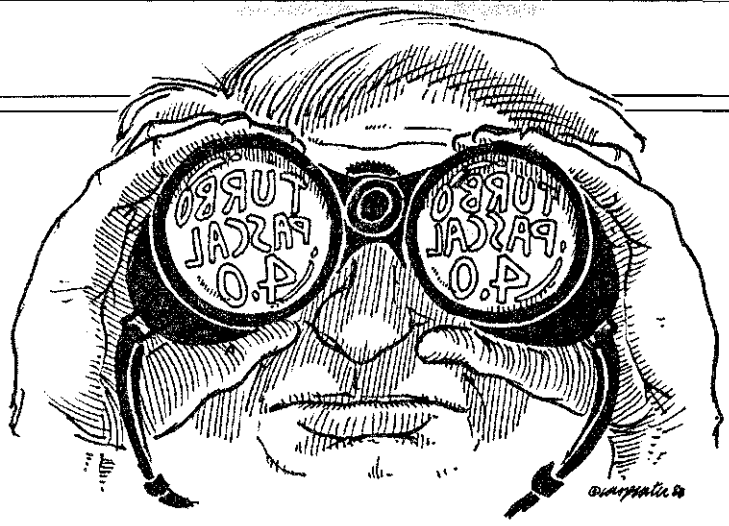
Turbo 4.0 Units

Each Turbo unit is organized as in the following example:

```
UNIT MyUnit;

INTERFACE
uses crt,dos;

TYPE
   Str80 = string;


VAR
   s80 : Str80;
   s : shortint;
   b : byte;
   i : integer;
   w : word;
   l : longint;
   r : real;
   p : pointer;

CONST
   c = 237;
   tc : longint = 123456;

function global(a,b : word) : word;

IMPLEMENTATION

TYPE
   local_str_type = string[20];

VAR
   local_i : integer;

CONST
   local_c : byte = 255;

function global;
begin
   global := a + b;
end;

procedure local(var str : local_str_type);
begin
   while length(str) >  5 do delete(str,length(str),1);
end;

begin
   w := global(7,8);
end.
```

This example illustrates just how a unit is constructed. Note first that each TP 4 unit has two major sections: an interface section and an implementation section. The interface section serves as an import, export manifest (note the functional similarity to Modula 2). A unit can import external program elements through the **uses** clause. In this example, the unit imports all the types, variables, constants, procedures, and

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



functions available for export from the declared units **crt** and **dos**. These are, by the way, Borland supplied units—more about them later. Similarly, the example unit exports all the constructs declared in the interface section. This means that these constructs will be available to any unit or main program that **uses** this unit. Notice that procedures and functions declared in the interface section are declared in a manner similar to standard Pascal forward declarations. The complete declaration, including the parameter list (if any) appears without the main body. The main body is declared in the implementation section and does not include the parameter list.

All constructs appearing in the implementation section, excepting those implementing interface declarations, are local to that unit and cannot be accessed from outside the unit. In the example, the function **global** may be accessed from any unit or program that **uses MyUnit** while the procedure **local** can be called only from within **MyUnit**. The same is true of the types, variables, and constants declared in the respective sections. One advantageous side effect of this scheme is that the programmer need not be concerned about the order of procedural bodies that are declared in an interface section since these are, in effect, forward declarations.

Another useful feature of the unit is the capability to include initialization code. Each unit must include a **begin end.** pair, just as in a main program. Any code placed there will automatically be executed when the main program is run. **MyUnit**, for example, is initialized by a call to the function **global**.

Turbo units are compiled separately to object oriented files with an extension designator .TPU . Thus, when compiled, **MyUnit** (if that is the name of the source file) will appear on disk as MYUNIT.TPU. The Turbo system will automatically link MYUNIT.TPU to any program that **uses** it. By the way, the Turbo linker is a "smart" linker in that it will link code only from those procedures/functions that are called from the using program/unit. This is an important efficiency since it means that programmers can create large unit libraries of useful routines knowing that at link time only the desired code will be placed in the target program. Borland includes a utility called TPUMOVER that performs rudimentary object library functions. The TP 4 intrinsic units—**System**, **Dos**, **Crt**, and **Printer**—are all located in a library file named TURBO.TPL. TPUMOVER allows the programmer to insert or remove compiled units from TURBO.TPL. Look for more flexible and powerful TP 4 library managers to emerge from third party commercial, public domain, and Shareware sources.

Intrinsic Data Types

As Tom Swan noted in his preview, a real strength of the new Borland compiler is the wide variety of Pascal data types offered. The unit example illustrates some of these. The standard TP 3 **integer** type (signed 16-bit **integer** and unsigned 8-bit **byte**) is included, but the general **integer** type is extended to include **shortint** (–128..+128), **longint** (–2147483648..+2147483647), and **word** (0..65535) types. The unsigned and long integers are particularly welcome additions. No longer does a programmer have to resort to real numbers when dealing with large integral values. Speaking of real numbers, TP 4 retains the 6-byte **real** from TP 3 and adds four additional **real** types for use on machines containing 8087 family numeric coprocessors. These are the **single** (4 bytes), **double** (8 bytes), **extended** (10 bytes), and **comp** (8 bytes). The **comp** type is comparable to a double precision **longint**, but is treated by the 8087 as a **real**. It is primarily suited for representation of monetary values as integral values.

Now that your appetite is whetted to feast on all those real types, let me tell you that this is one of the few deficiencies in the TP 4 system.

Now that your appetite is whetted to feast on all those **real** types, let me tell you that this is one of the few deficiencies in the TP 4 system. TP 4 does not emulate the 8087 with software. The coprocessor must be present in the machine upon which the code is compiled, and that code will execute only in machines that contain the coprocessor. This means that the additional **real** types are essentially useless to anyone who wishes to develop software targeted to run on the widest variety of PC family machines. If the programmer doesn't have a machine with a coprocessor, he can't use any but the standard Turbo 6-byte **real**. I strongly urge Borland to rectify this situation by providing 8087 emulation in the compiler.

Two other data type variations from TP 3 deserve mention. The TP 4 **string** type defaults to **string[255]**, and the new compiler implements a generic pointer type named, strangely enough, **pointer**. The **pointer** type can point to anything and is initialized to nil (0000:0000).

Continued . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Overlay Manager Borland Forgot

I'm not alone in my feeling that even though Turbo Pascal 4.0 supports a large code model, the compiler should have an overlay capability. TurboPower Software has introduced Overlay Manager (OM) 4.0, a complete package that offers both overlays and chaining for TP 4. TP 3 programmers familiar with TurboPower's excellent Turbo Optimizer and Turbo Extender will not be surprised to learn that OM 4 is a quality product.

Not only does TurboPower's overlay system completely support the creation and management of code overlays for TP 4, but it also offers two dynamite features not available in TP 3 overlays. The first is that a procedure or function in an overlay file may call another procedure or function in the same overlay. The second is that, with no additional work required of the programmer, OM 4 will automatically place the overlay code in expanded memory, provided that a resident EMS driver is detected and reports sufficient memory pages available to contain the code.

There are four relatively straightforward steps to follow in order to build a TP 4 overlay application with OM:

1. Import the supplied OVERMGR.TPU unit by listing its name in the uses clause of the target program. Compile the program using the {$T+} option to create a TP 4 .TPM file.

2. Use the TP 4 utility TPMAP to convert the .TPM file to a .MAP file.

3. Select the units to overlay, and place their names in an ASCII control file with the extension .OSP.

4. Run TurboPower's overlay processor, OVERLAY.EXE.

I processed several programs, including one large TSR model, and found that Step 3 required the most thought because that's where the programmer actually designs the overlay structure. OM organizes overlays into groups (6 groups maximum), and each group may contain up to a whopping 51 units of code. One caveat is in order. If the main program uses any heap space and sets its size to less than maximum, the programmer will probably have to do a recompile. This is because OM requires some dedicated heap space and the total needed is not known until determined (and reported) by OVERLAY at runtime. 400 bytes are dedicated to the OM stack, and the remainder is used for a list of TP 4 address fixups, which cannot be known until OVERLAY processes the TP 4 code. TurboPower also provides instructions (and an example) for creating a batch file that uses the command-line version of TP 4 to rebuild an application.

The only limitation of OM is that certain TP 4 units may not be overlayed. These are the **System** and **Crt** units, any unit that contains interrupt handlers, or units with routines that are called indirectly via pointers. OM reports the successful creation of TP 4 executable files larger than 1 megabyte. One important plus is that the programmer can change the program's overlay structure without modifying source code or even recompiling.

During program execution, OM will look for the overlay file in two places: the current directory and, if DOS 3.x is present, the directory of the .EXE file. TurboPower also provides source code for an installation program that permits setting the path name of the directory where the overlay file will reside.

My tests of OM showed no problems. I was pleased to note that I was able to run a program with overlays in expanded memory at the same time that a popular disk cache program was active in expanded memory. The code in the overlay performed a lot of disk I/O, and the fact that there were no problems shows conclusively that the folks at TurboPower know what they're doing. Global variables and constants that give the programmer the option of preventing use of expanded memory and control over the dedicated OM stack are declared in OVERMGR.TPU.

The TurboPower package also includes a utility that implements TP 3 type chaining in TP 4—although with overlays and everything else TP 4 does, I don't see the efficacy of chaining. The TurboPower package is augmented with TP 4 source code to the supplied unit—a nice touch. All in all, this is an excellent package, and I highly recommend it to those whose TP 4 applications require an overlay capability.—*John J. Newlin* ♦

Overlay Manager 4.0
TurboPower Software
3109 Scotts Valley Drive, Suite 122
Scotts Valley, CA 95066
(408) 438-8608
$45.00

The Programming Environment

As user friendly as the TP 3 programming environment was, TP 4 provides a system environment that far outshines it. This is one of the major strengths of this compiler. The TP 4 environment stays out of the programmer's way and eases the development and management of large, complex programming projects. Moreover, Borland has provided the programmer with the means to tailor the development environment to his or her needs and/or tastes. Space limitations preclude my giving you detailed descriptions of all the system features provided in TP 4, so I will attempt to highlight those that are the most utile.

TP 3 users will appreciate the fact that the look and feel of the integrated editor has been retained in TP 4. TP 4 is a complete development system, however, in which the editor is just one component. The system is controlled through a nicely implemented hierarchy of pull-down and pop-up menus. Source and object file handling is a dream! A Pick menu provides the means by which you can quickly select and load source files. One really nice feature is that the TP 4 system "remembers" the last cursor position of each file so that when you reload a file, you are viewing the same block of code that was in view when the file was last saved. I only wish that Borland had provided a split-screen, dual file editing capability that supported quick transfer of source text blocks between files, but that would be sweet icing on an already delicious cake. Online help is extensive (150 screens), and one feature available while in the editor merits special mention. If you cannot remember the specific syntax for a procedure or function declaration, it is not necessary to dig out the rather bulky (654 pages) user's manual. Simply place the cursor somewhere on the identifier and press Alt-F1. A window will pop up that contains the correct declaration, a brief functional explanation, and a reference to related procedures/functions. Now, that's nice!

Compiling to and running from dynamic memory is fully supported, and if an error is detected in one of the source files, that file is loaded and displayed with the cursor positioned at the offending source text. The linker is normally resident in memory with the compiler, but if host system RAM is limited, you can specify that the linker be called from disk. Compiler directives may be controlled through the Options menu or specified, a la TP 3, as source statements in the form {$-V}. Active compiler directives can be inserted into the source file at the current cursor position by pressing Ctrl-F7. This is particularly useful if you intend to use the command-line-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

operated version of the compiler (also supplied). Stack and heap size settings can also be set from the Options menu or via source code compiler directives.

TP 4 Pascal is a single-pass compiler, but it does perform several types of code optimizations, including constant folding/merging, short-circuit Boolean evaluation, shift instead of multiply, and dead code removal. The Options menu provides the means for turning range, stack, and I/O checking and debug information switches on or off. I compiled a fairly large program with all checking on and then recompiled it with all checking off. The savings realized by the absence of the code necessary to support the checking was a whopping 15K! It pays to turn all the switches off once the program is completely tested and debugged.

Programmer control of the type of compile/link operation desired is accomplished from the Compile menu. A Build operation will compile and link all units declared in the target source files. A Make operation will recompile only those source files that have changed since the last compile. Borland supplies a separate Touch utility, TOUCH.COM, used to update the date/time stamp of a file. TP 4 also supports the Find error facility pioneered in the original Turbo Pascal, but the address is now specified in full segment:offset address form, necessary because each TP 4 unit resides in a separate code segment. A very nice feature available from the Compile menu is a pop-up window that displays data pertinent to the last compile operation performed. This information includes source file size, lines compiled, and code, data, heap, and stack sizes.

One last comment on the TP 4 environment is in order. I personally detest command-line-oriented compilers and linkers. TP 4 provides a powerful integrated system as an alternative. Combination compile and link speeds are blazing fast, and source file handling is quick and easy. The capability to alternately code and test by compiling to and executing from memory is indispensable. It would be very difficult to create a more efficient development environment. What all this means is that the programmer is free to concentrate on employing his or her creative talents in developing the best possible program and not forced to expend valuable time and effort in coping with an unfriendly environment.

Compatibility with Turbo 3.0

TP 4 is designed around a completely different architecture than that of TP 3, and therefore it would not be surprising to encounter major problems when porting TP 3 source code to TP 4. But that, happily, is not the case. The TP 4 manual devotes an entire chapter to the differences between the two compilers, and the system includes a utility named UPGRADE that operates on TP 3 source code. The information provided in the TP 4 manual was so complete that I was able to port several TP 3 programs with no major problems without using UPGRADE. Borland also includes two compiled units, **Turbo3** and **Graph3**, that facilitate the compilation of TP 3 code.

The ever popular **inline** code facility of TP 3 is supported in TP 4. The only problem I encountered with inline code was in porting the following TP 3 line:

```
$E8/>HANDLER-*-2      {call >HANDLER}
```

TP 4 would not compile this line since it supports neither the location counter reference (*) nor procedure/function identifiers in inline code. The way around this is to initialize a pointer to the desired procedure and then reference the pointer identifier in the inline code. TP 4 extends the inline capability by providing the means to write short inline procedures that are inserted as macros wherever the call appears.

One difference that cropped up often in my source code was the use of the TP 3 **Read(Kbd)** procedure. TP 4 does not support **Read(Kdb)** and substitutes the much more useful **ReadKey** function in its stead. Another difference will probably surface often when porting TP 3 to TP 4: The familiar 8088 register definition record used with the **Intr** and **MsDos** procedures is predefined in the TP 4 **Dos** unit, necessitating removal of the **RegPack** record type declaration from TP 3 source code. Typed constants are supported by TP 4, but with one major, and perhaps disconcerting to some, difference. In TP 3, typed constants (not really constants at all) resided in the code segment, where they were readily available for use by interrupt handlers and other system oriented constructs. In TP 4, typed constants reside in the data segment.

TP 3 provided no intrinsic interrupt handling directives or constructs. TP 4 includes procedures for getting and setting interrupt vectors as well as an intrinsic procedure for terminating and remaining resident. Also, an **Interrupt** declaration is provided for designating a special interrupt handling procedure. This procedure may have a specified list of **integer** type parameters corresponding to the registers automatically saved as a result of the **Interrupt** declaration. This allows you to access and/or set the registers without having to resort to assembly or inline code. Another function ac-

SRMS™

"...is the easiest to use of those [Revision Control Systems] with screen based menus..." PC Tech Journal Sept '87

"...is a practical and versatile program that many programmers will want to use daily" IEEE Software July '87

SRMS is a full featured revision control system that allows programmers, system managers, and librarians to get control of software projects. SRMS stores every version of a source program in a single ASCII file, with the ability to restore each and every version easily and quickly. Only changes (deltas) between versions are actually stored, so disk usage is drastically reduced. Each version is stored with a unique Version Identification String and unlimited user supplied comments, making identification and documentation tasks easy. SRMS supports projects written in any language, and allows you to use your current editor and compiler without conflict. Version MERGE facilities help consolidate parallel development efforts. SRMS also provides the ability to bind various versions of different modules together for reconstructing a particular software system release. DOS directory pathnames and environment variable support also aid you in your work. Full typeset documentation with tutorial and project management chapters.

Version 3.1 Now Available
- Up to 50% faster
- Enhanced Version Numbering
- Configuration File Support
- Comment Line Editing
- Library Module Extension Control
- Increased Wildcard Handling

Multi-User SRMS Now Available
- Same Features of Single User SRMS
- Journaling and Authorship Facilities
- Library Access Control Lists
- LAN Compatible

SRMS Integrates Fully with QMAKE

QMAKE is a powerful and flexible utility patterned after UNIX make, that can be used to control the rebuilding of simple or complex software systems. QMAKE is able to use the date/time stamps of source stored with SRMS to rebuild your systems quickly and accurately.

Text Manipulation Tools

TXTTOOLS is a set of three utilities that allow you to efficiently search for, edit and show the differences associated with text files.

Single User SRMS	$185
Multi-User SRMS	$250
SRMS 5 Site LAN	$750
QMAKE	$99
SRMS+QMAKE	$250
TXTTools	$85

30 Day Money Back Guarantee

BUILT COMPUTING
7048 Stratford Road
Woodbury, MN 55125
(612) 739-4650

Volume Discounts and Dealer Inquiries Welcome

Number 46 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

complished by such a procedure is to automatically initialize the DS register to the program's data segment. All of this is well and good, but I have a problem with the manner in which this procedure is implemented. By default, all of the registers are saved on the stack upon entry, before the programmer gets control, and popped from the stack upon exit. Pushing and popping eats a lot of time (11 clock cycles for a push, 8 for pop), and if the handler is time dependent, such as when servicing the timer interrupt, a lot of time is wasted pushing and popping registers that may not require saving. I would have preferred more programmer control, but Borland deserves an "attaboy" for including this facility as an intrinsic construct.

Other differences in the two compiler versions exist, but they are clearly identified in the TP 4 manual. The bottom line, in my opinion, is that most TP 3 source code is, with a little work, readily portable to TP 4, and, given the huge base of that code, that's good news for Turbo Pascal programmers.

Assembly Language Interface

With two very minor exceptions, the assembly language interface is another area in which TP 4 really shines. Intel format object code produced by an assembler is readily linkable to TP 4 code, and both near and far call modes are supported. The Borland owner's manual, while not effusive, gives adequate explanation and examples of how to implement the interface to assembled modules. The TP 4 linker will not initialize data segment variables (i.e., **my_var db 123**) but instead requires the question mark (?) to specify variable values. Nor does the linker operate in the "smart" mode when linking external object modules. For example, if you have an object module containing ten assembly procedures, but refer to only one of them in the TP 4 source code, the entire module is still inducted by the linker.

Turbo procedures and functions can be called from assembly modules, and Pascal variables can be referenced as externals as well. By the way, I was most pleased to see that Borland acknowledges Eric Isaacson's great A86 assembler in their manual. Version 3.11 of A86 fully supports linkage to TP 4 programs, and I highly recommend it for its speed and unique features.

Compiler Performance

As advertised, TP 4 fairly flies during compilation. It operates at the speed of heat on a 10 MHz 286 machine. During compilation, the programmer is kept advised of the file being compiled, current line number, and total lines compiled. I think that executable code quality (speed and compactness) is more important than compiler speed, but the high compilation speed of TP 4 will sure save a lot of the programmer's time over the course of project development.

My view is that compiler benchmark data should be examined with caution by those evaluating compiler performance. Benchmarks are useful, however, in gauging the quality of the executable code produced by a compiler. Unfortunately, PC benchmark metrics are not standardized and vary widely from review to review. The benchmark data presented here was derived from exercising the tested compilers on an AST Premium/286 running at 10 MHz. I thought it would be interesting and useful for readers to see comparative data for three different Turbo Pascal compiler outputs. Of course, TurboPower's Optimizer is not a version of TP, but I included it because it is used by many serious TP programmers. Results from the Marshal Language Systems compiler are included because there are few compilers of any language for the PC that produce comparably fast code.

Benchmark Results

	TP 4		TP 3		TP 3 OPT		MLS	
	Time	Size	Time	Size	Time	Size	Time	Size
Sieve	10.2	6648	11.5	12062	9.7	4414	4.0	9516
Ackerman	18.1	6352	24.5	11965	18.9	4244	16.8	9412
Float 4	N/A	N/A	N/A	N/A	N/A	N/A	1.4	9422
Float 6	3.9	6488	5.0	11976	5.0	4326	N/A	N/A
Float 8	N/A	N/A	N/A	N/A	N/A	N/A	3.3	10414
Puzzle	8.2	8832	9.0	15343	7.1	6982	6.6	11136
Null	N/A	1360	N/A	11433	N/A	2038	N/A	2032

These results were obtained by executing the benchmark programs on an AST Premium/286 running at 10 MHz. TP 3 OPT is code executed after modification of Turbo 3.0 code by TurboPower Software's Turbo Optimizer. MLS is code generated by the Marshal Language Systems Pascal compiler. All source code contained two **Writeln** statements and a timing routine. All compiles were made with appropriate compiler directives set for maximum optimization and minimum code checking. The number following the category Float indicates the size, in bytes, of the type **Real**. The Null results were obtained by compiling an empty program, i.e., **begin end.**, and indicate the size of the initialization code produced by the compiler. Compile times were not included due to the rather small sizes of the benchmark source files. Suffice it to say that Turbo 4.0 compiles source code several times faster than any other Pascal compiler.

The source code for these benchmark programs is available on the Programmer's Journal *Issue Disk for Volume 6.3 and on PJ's bulletin board services.*

Debugging Support

No debugger is provided with the TP 4 package. Borland has designed its symbol table and MAP file format to be compatible with the commercially available Periscope debugger, and a short chapter in the user's manual describes how to use Periscope with TP 4. To me, this is just further evidence that Borland views TP 4 as an end user product and not as a commercial software development system. That's too bad because if Borland had integrated a quality symbolic debugger into the TP 4 environment, included 8087 software emulation, and provided overlay support, it would have a compilation system that would blow Microsoft C right out of the water. I would much rather manage a large code development project written in TP 4 than I would one written in C. Pascal is much more readable than C (someone once said that C is a write-only language), and TP 4 units are perfect for a modular design approach.

Final Observations

TP 4 units may each contain up to 64K of code, thus breaking the 64K code barrier imposed with TP 3. But TP 4 supports just one data segment, forcing the programmer to go to the heap if more than 64K of data is required. It seems to me that a future release might support separate local data segments for each unit as well as a global data segment available to all units.

No intrinsic support is provided for creating programs that use the new LIM 4.0 Expanded Memory Specification (EMS). I believe that EMS is going to play an ever increasing role in the future of PC software. No compiler currently available provides direct EMS support. Someone at Intel Cor-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

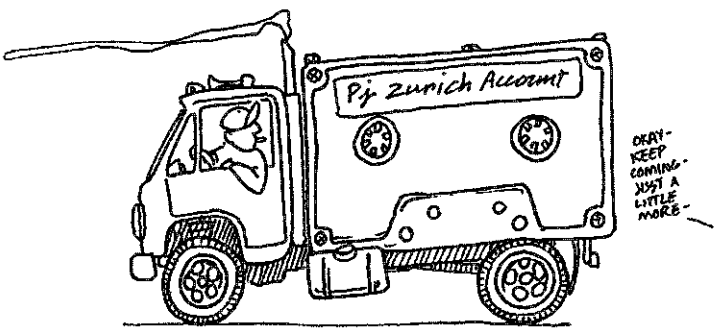
poration recently observed to me that, in his opinion, exploitation of the EMS has been deterred by the lack of EMS programming tools.

Give Borland another "attaboy" for the inclusion of direct screen display writing via the **Write** and **Writeln** procedures in the **Crt** unit. If desired, the programmer can select the more normal convention of video BIOS calls. Selectable checking for CGA video retrace is also included when using direct screen writing. One nit to pick here is that even in the direct screen writing mode, TP 4 advances the cursor one column after a **Write** operation. This causes a screen scroll-up if a character is written to column 80, row 25 of the video display.

TP 4 offers no support for overlays. It might be argued that since TP 4 allows greater than 64K of code, overlays are not needed. I think many programmers will disagree for a variety of reasons. I have a TP 3 memory resident application that relies heavily upon overlays to minimize the amount of code resident in RAM. Without overlay support, I can't port the TP 3 code to TP 4. Once again, Kim Kokkonen's TurboPower Software comes to the rescue. TurboPower distributes an overlay manager for TP 4 that not only does the job, but will also automatically place the overlay in expanded memory if it is available. [See the sidebar on TurboPower's Overlay Manager.]

Gone is the convention of using **Write(Lst)** to access the printer. TP 4 provides a separate unit named **Printer** that must be declared in any unit that writes to the printer device. TP 4 provides a number of useful predefined variables and constants in the packaged units. One of these, **PrefixSeg**, contains the segment address of the Program Segment Prefix (PSP). In the case of TP 3 .COM files, it is easy to access the PSP since DOS places it at offset 0 of the code segment. But it can be very difficult to locate in .EXE programs that contain multiple segments. Thank Borland for thoughtfully capturing the PSP segment value for us at runtime.

Extensive graphics support is included in the TP 4 package. The **Graph** unit contains a library of more than 50 graphics routines, including bit-level routines that save and restore graphics images. Graphics drivers that support different graphics cards are supplied in binary files with the extension .BGI. At runtime, a **Graph** unit routine will automatically identify the type of graphics hardware in the host system, load and initialize the appropriate driver, and place the system in the graphics mode. Drivers are included for MDA, CGA, EGA, VGA, Hercules, AT&T 400 line, and 3270 PC graphics adaptors. Three character fonts are also included. Borland permits distribution of these files pursuant to the terms of their TP 4 licensing agreement.



— TAPE BACKUP —

TP 4 is clearly a winner! I couldn't find any bugs in the compiler or any problems with the integrated environment. I have already ported one of my Shareware products to TP 4 and am looking forward to designing and coding new projects. If you are a TP 3 programmer and haven't upgraded, I strongly urge you to do so. At $40, it's a steal. If you are a programmer who appreciates the structured elegance and modularity of Pascal and needs the power and flexibility of separate compilation, give TP 4 a try. You'll like it. ♦

Turbo Pascal 4.0
Borland International
4585 Scotts Valley Drive
Scotts Valley, CA 95066
(408) 438-5300
$99.95 (upgrade $39.95)

John J. Newlin is a retired Naval Aviator who currently works for Hughes Aircraft Co. as a systems engineer. John holds degrees in mathematics and computer science and has programmed extensively in assembly, Turbo Pascal, Marshal Pascal, and Modula 2. He is the author of SCOUT, a memory-resident disk/file/directory manager (reviewed by PC Magazine, *February 24, 1987), and ARCMASTER, a "point and shoot" style interface between the user and ARC and de-ARC programs (reviewed in* Computer Shopper, *January 1988, recently ported to Turbo Pascal 4.0). John is a charter member of the Association of Shareware Professionals. Readers may write to John at the following address:*

John J. Newlin
4060-228 Rosenda Court
San Diego, CA 92122

TURBO
SOURCE SEARCH

TSS is a programmer's SIG that currently has 240+ disks representing 80MB of archived files!! TSS specializes in locating publicly accessible files in Turbo Pascal, BASIC, QBASIC, C, QuickC, ASM and Modula-2. We offer these files by mail as well as on-line.

TSS-BBS (617-545-9131 within the PC-Pursuit area) is an outlet for the programming code from *Programmer's Journal*. All files in the TSS SIG are freely accessible through the BBS to members and non-members. Members have increased time and access. A central library for publicly accessible programming utilities is a valuable resource, and we are certain that programmers recognizing that will support our efforts by becoming members.

TSS is also a BBS-by-mail. All files maintained by TSS-BBS are available to members for as low as $2.50/disk. All files on all disks are archived to maximize the number of bytes per disk. A yearly membership to TSS, $40.00, entitles you to purchase disks from the library for $4.00/disk. Bulk discounts can lower the price to $2.50/disk. We also offer custom filled disks. Send $4.00 for the catalog disk or $3.00 for a printed catalog. (Members receive first catalog disk FREE.)

TURBO SOURCE SEARCH
P.O. Box 11, Scituate, MA 02066
Voice: 617-545-6677
Data: 617-545-9131, 1200/2400 BAUD, 24 hrs.
VISA/MC/Check/Money Order/non-US extra

Number 35 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Introducing a whole new world in C performance.

C6.0

WATCOM unleashes a high-performance optimizing C compiler and development system that delivers superior results for every professional programmer and offers the fastest programming environment you can get your hands on.

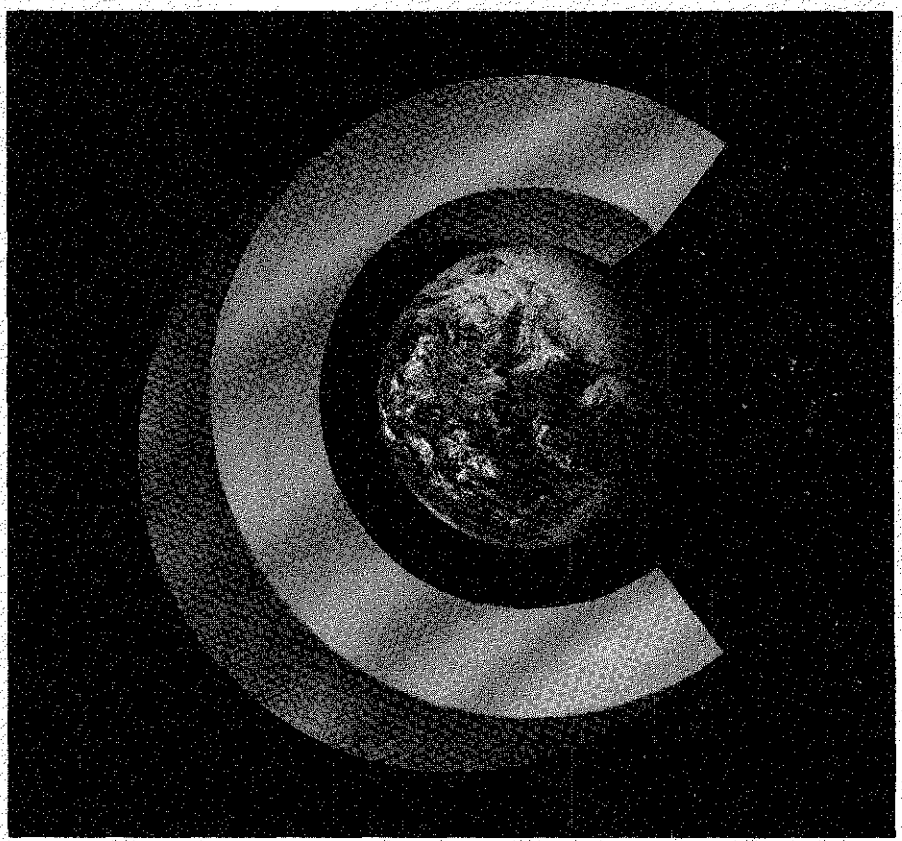
Producing the fastest, tightest code available today, WATCOM C6.0 offers major advantages in large memory models and floating-point computation.

The WATCOM C6.0 system includes WATCOM Express C (also sold separately), a seamless development environment complete with unexcelled diagnostic capabilities to maximize your productivity.

World Class Performance

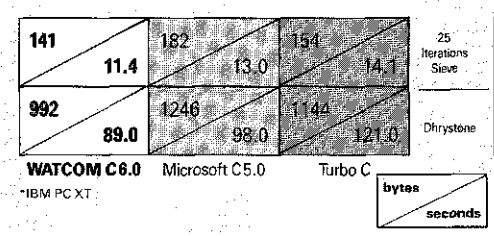
Benchmark tests pitting WATCOM C6.0 against Microsoft C5.0 and Turbo C demonstrate the optimum code size and speed your programs will achieve with WATCOM C6.0. Sophisticated register allocation, true register variables and flow analysis allow your code to run its fastest. Moreover, extensive fine-tuning is possible with numerous user options like in-line substitution of machine code for performance-critical areas.

We deliver more than performance. Flexible run-time conventions allow efficient interfacing to a wide range of libraries and language processors. WATCOM C6.0 supports small, medium, compact, large and huge memory models and the NEAR, FAR and HUGE keywords.

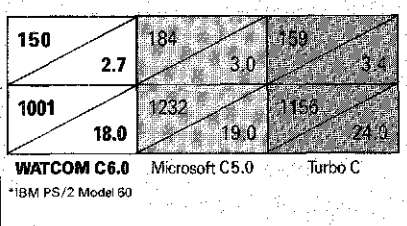


Announcing the world's fastest, tightest code.

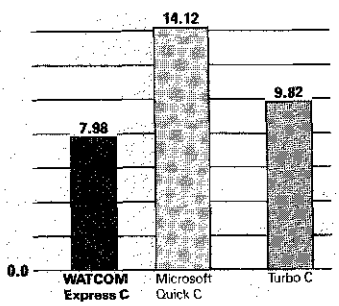
The fastest, tightest code
(small memory model*)



The fastest, tightest code
(medium memory model*)

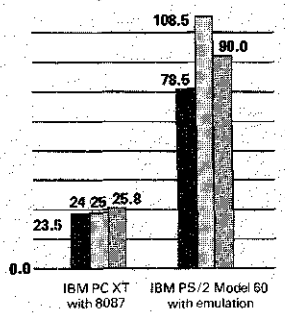


Turnaround Time (compile + link/load)



Time in seconds to compile 573 source lines plus includes, and link 7 additional OBJ files created from 1391 source lines. IBM PS/2 Model 60, small memory model. Program used: GREP.

Floating point computation



Time in seconds to run 25 iterations of Whetstone. Small memory model (64K code, 64K data)

Number 181 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Maximum Productivity

Express C's fastest available compile turn-around assists you with syntax and prototyping. With integrated tools like an editor and a symbolic debugger, you'll be amazed at how quickly you see your results.

The Right Tools for the Job

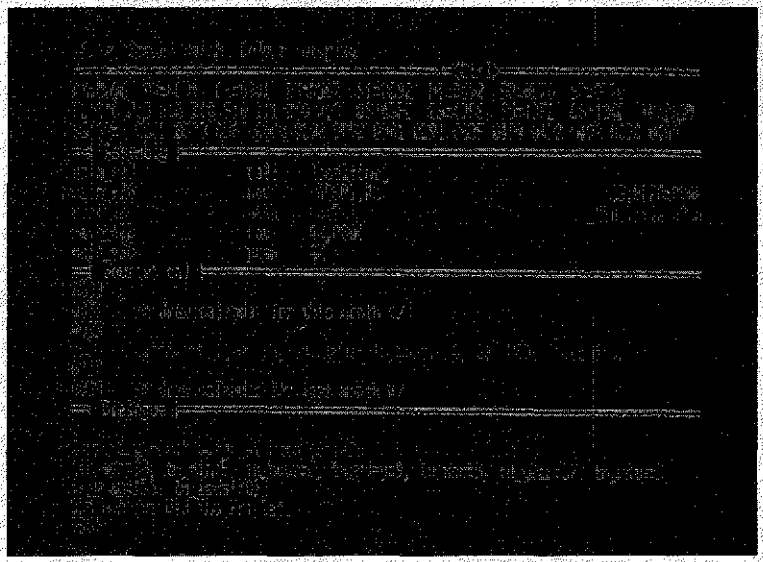
Express C, included with WATCOM C6.0, has unique error checking that quickly uncovers many common and difficult bugs in "correct" code that other compilers miss. Take advantage of the powerful integrated debugging in Express C, then use WATCOM C6.0 for even tighter code and faster execution speed. Comprehensive compile-time and run-time messages enable effective programming practices which you can extend with our fast development tools: **MAKE**, a **disassembler**, an object **librarian** and an overlay **linker**.

Technical Support

Because we know how valuable your time is, our on-line help is always available. Our software maintenance and licensing plans will keep you up to date. With them, our expert technical support is just a phone call away. When you create a great new product, our run-time library code can be licensed without royalties or accounting. Naturally, training seminars are available at our Customer Education Center or your site.

Superior Value

The WATCOM C6.0 system delivers superior value, returning your investment promptly by saving your valuable time. Our twenty years of computer language experience began in 1965 when our **WATFOR** authors wrote the book on high-productivity debugging compilers. The WATCOM C compiler is based on the sixth in a series of 8086-family code generators used since 1981. Today, we remain committed to delivering world class programming tools.



With WATCOM C6.0, you can debug large applications without extended memory.

WATCOM C6.0

Superlative Performance

- Full ANSI C Optimizing Compiler
- Source Editor
- Full-screen Source-level Debugger
- Full ANSI C Run-time Library
- Overlay Linker
- Object Librarian
- MAKE
- Disassembler
- **Express C**

List price: $495

Introductory Price: **$295***

WATCOM Express C

Unparalleled Productivity

- Full ANSI C Compiler
- Integrated Source Editor
- Integrated Debugger
- Full ANSI C Run-time Library
- Integrated Linker/Loader
- Overlay Linker
- Object Librarian
- MAKE

List price: $125

Introductory Price: **$75***

Available for IBM PCs, PS/2s and true compatibles with DOS. General availability March 1, 1988.

*Limited time introductory prices apply only to prepaid orders (VISA/MasterCard). Shipping and handling extra.

WATCOM

_____ I want to order **WATCOM C6.0**, regularly priced at $495 and well worth it, for just **$295**.

_____ I want to order **WATCOM Express C**, regularly priced at $125 for just **$75**.

_____ Please send me organization site licensing information.

_____ Please convince me further by sending a complete product description of WATCOM C6.0.

Name _____ Title _____

Company _____

Street _____

City _____ State _____ Zip _____

Information needed for Credit Card orders:

Signature _____ Tel. _____

Credit Card: Visa _____ MasterCard _____

Card # _____ Expiration Date _____

WATCOM
Dept. PJ-03
1430 Massachusetts Avenue
Cambridge, MA 02138

For even faster service call:
1-800-265-4555

WATCOM C and **WATCOM Express C** are trademarks of WATCOM Products Inc. ©Copyright 1988 WATCOM Products Inc.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

EVEN MORE POWER AND FLEXIBILITY

BRIEF 2.0

Users and industry press alike have unanimously proclaimed BRIEF as the best program editor available today. Now the best gets better with the release of BRIEF 2.0.

Straight from the box, BRIEF offers an exceptional range of features. Many users find that BRIEF is the only editor they'll ever need, with features like real multi-level Undo, flexible windowing and unlimited file size. But BRIEF has tremendous hidden power in its exclusive macro language. With it, you can turn BRIEF into your own custom editor containing the commands and features you desire. It's fast and easy.

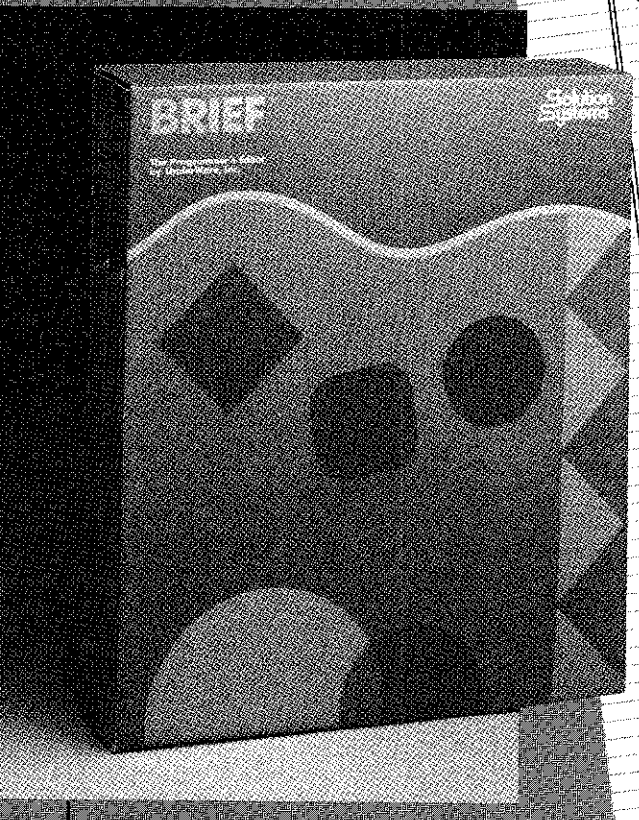
Jerry Pournelle, columnist for BYTE magazine summed it all up by saying BRIEF is, "Recommended. If you need a general purpose PC programming editor, look no further." His point of view has been affirmed by rave reviews in C JOURNAL, COMPUTER LANGUAGE, DR. DOBB'S JOURNAL, DATA BASED ADVISOR, INFOWORLD AND PC MAGAZINE.

One user stated "BRIEF is one of the few pieces of software that I would date call a masterpiece." Order BRIEF now and find out why. BRIEF 2.0 is just $195. If you already own BRIEF, call for upgrade information.

TO ORDER CALL: 1-800-821-2492 (in MA call 617-337-6963)

As always, BRIEF comes with a 30 day money back satisfaction guarantee.

Solution Systems™
341 Main Street
Suite 410
So. Weymouth, MA 02190
(617) 337-6963



Look at these BRIEF 2.0 enhancements!

Main Features:

- All new documentation with tutorials on basic editing, regular expressions and the BRIEF Macro Language.
- Setup program for easy installation and configuration. (Requires no knowledge of the macro language)
- Increased speed for sophisticated operations like Undo and Regular Expression Search.
- Expanded regular expressions, with matching over line boundaries.
- More block types, with marking by character, line or column.
- Command line editing (move cursor, add and delete characters, specify command parameters).
- Support for more programming languages.
- Optional borderless windows.
- Enhanced large display support, including wider displays.
- Reconfigurable indenting for C files (supports most indenting styles).

Plus the basic features that made BRIEF SO popular!

Basic Features:

- Full multi-level Undo
- Windows
- Edit many files at once
- File size limited only by disk space
- Automatic language sensitive indentation

Requires an IBM PC or compatible with at least 128 RAM.
BRIEF is a trademark of UnderWare Inc.
Solution Systems is a trademark of Solution Systems.

Number 47 on the Reader Service Card.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Proposed ANSI C Language Standard

by
Rex Jaeschke

© *Rex Jaeschke, 1987. All rights reserved.*

In this and the installment in our next issue, Rex takes time out to respond to a reader's questions and comments about the emerging ANSI C Language Standard.

I recently received a letter from David R. Tribble of Dallas, Texas. David's letter contained a number of interesting questions and comments about the proposed ANSI C Standard, and I've chosen to include some in this column.

David's comments are relevant to the draft of October 1, 1986 (Doc. No. X3J11/86-157)—the current public version. Committee members, however, get an updated draft every three months, so my most recent version is dated August 3, 1987.

David's points and my responses follow.

Question 1. The NULL macro [3.2.2.3, 3.3.6, 4.1.4]
Shouldn't the macro NULL in **<stddef.h>** be a constant of type **void ***? In all the implementations I've seen, NULL in **<stddef.h>** is always **#def**ined as **0** or **0L**. The reason this is important is illustrated in the following fragment:

```
int *p;
... (p - 0) ...          /* 1 */
... (p - NULL) ...       /* 2 */
```

According to [3.3.6], Expression 1 has value **p** and type **int ***. If NULL were indeed **#def**ined as **0** or **0L**, an int constant, Expression 2 would also result in an **int *** of value **p**. If, on the other hand, NULL were declared as a **void *** *constant*, Expression 2 would evaluate to an integer value of type **ptrdiff_t**. Granted, the above situation is not very common, but it could occur as part of some macro the programmer created assuming that the types would take care of themselves.

While [4.1.4] states that NULL expands to an "implementation-defined" null pointer constant, the argument could be made that since the int constant **0** is a special case of a pointer constant, as stated in [3.2.2.3], **0** or **0L** is acceptable as the replacement value for NULL even though it is not of a pointer type; thus, in implementations that define NULL as a constant integer expression with a value of zero, the situation above could arise.

My reply: The draft currently states, "An integral constant expression with the value **0**, or such an expression cast to type **void ***, is called a null pointer constant. If a null pointer constant is assigned to or compared for equality to a pointer, the constant is converted to a pointer of that type. Such a pointer, called a null pointer, is guaranteed to compare unequal to a pointer to any object or function." The Standard, therefore, currently allows NULL to be defined as any one of **0**, **0L**, and **(void *)0**.

I believe we agree that the meaning of Expression 1, above, is quite clear—however, your discussion of Expression 2 is somewhat erroneous. You correctly state that Expression 2 behaves as Expression 1 does if NULL is either **0** or **0L**—however, if NULL is equivalent to **(void *)0**, you should get an error.

C discusses (and promises certain behavior for) the subtraction of two pointers only in the very specific case where both pointers point to elements in the same array of objects. This means that both pointers must have the same type, and clearly, **int *** and **void *** are not the same type even though these types are assignment-compatible.

Note that **p** and **(int *)(NULL)** do have the same type, and their difference is of type **ptrdiff_t** as you suggest. But this produces undefined-behavior since the two pointers do not point to members of the same array.

Another subtle problem is that the draft defines pointer subtraction only in terms of two "pointers to objects," and **void *** is not a pointer to an object—it is a generic pointer that may contain the address of an object. Until that pointer is cast to some other pointer type, it cannot be considered a pointer that designates an actual object.

Part of the problem is that for many years C programmers have (intentionally or otherwise) mixed integers and pointers—they "knew" that NULL was an integral constant with value **0**, so they (mis)used NULL when an integral expression was actually required. NULL means, and has always meant, the null pointer; therefore, I suggest that Expression

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

2, above, has always been "incorrect," or, at least, inappropriate.

As far as I can tell, K&R did not require that NULL be defined as **0**; an implementer could well have used **(char *)0** instead to achieve the same result. Nowhere has this been more appropriate than on 8088-class segmented machines where pointers can be 16 or 32 bits depending on the compilation model. If **(char *)0** were used for NULL, the messy conditional compilation into **0** or **0L** depending on model, as used by popular DOS compilers, could be eliminated.

Question 2. Void pointer arithmetic [3.3.6]
Consider the following code fragment:

```
int *pi;
void *pv, pv2;

... (pi - pv) ...     /* 1*/
... (pv - pv2) ...    /* 2 */
```

Expression 1 will evaluate to an **int** value of type **ptrdiff_t** equal to the difference between **pi** and **(int *)pv**. But what about Expression 2? Shouldn't it be illegal to perform arithmetic with void pointers? In other words, is the size of the object pointed to by a void pointer zero, or does it make more sense to say that a void pointer doesn't point to anything and that since an array of void objects is prohibited, the difference between two void pointers is meaningless? I ask this question because I can't find any mention that arithmetic with void pointers is illegal or meaningless.

What about pointers to functions in this context?

My reply: Most of these questions are answered in 1, above. Both expressions are invalid. The whole purpose of **void *** is to provide a "knothole" through which you can stuff a pointer of any type and get exactly the same thing out the other side. A pointer to void has only a value; it has no attributes and therefore can't take part in any context where attributes are needed.

As discussed in 1, above, with subtraction both pointers must point to objects, and **void *** does not. And even if **void *** pointers were allowed here, the answer would be meaningless since the difference between two pointers is the difference between their values divided by the size of the object they point to. What is the size of the object pointed to by a **void ***? As it happens, **sizeof(*pv)** cannot be determined and should result in an error.

Pointers to functions cannot take part in arithmetic, and **sizeof(*functptr)** should produce an error.

Question 3. Switch statement body [3.6.4.2]
Not much is indicated about the content of the body of a switch statement, specifically local declarations. According to K&R [A.9.7], the switch statement body may contain declarations, but auto and register initializations are "ineffective." No such restriction is mentioned in the draft, and [3.6.2] implies that there isn't any. Is this true?

A complex (and horrendous) example of a switch body is

```
switch (st) {
case 1:  st = 1;  do {
case 2:  st = 2;     printf("%d\n", c);
case 3:  st = 3;     c = getchar();
case 4:  st = 4;  } while (c != EOF);
default: st = 5;
}
```

The obvious response one has to this sort of code is "Why? There's got to be a better way!" But this sort of situation arose in a program that could be interrupted at any point and then resumed at the point where it left off, represented in the fragment above as state variable **st**. My question: Under the stan-dard, will this work? As I said before, the Standard doesn't say much about what is allowed or disallowed within switch statement bodies.

My reply: The formal grammar for switch has always been

```
switch (expression) statement
```

As stated in K&R (page 203), "Usually the statement that is the subject of a switch is compound. Declarations may appear at the head of this statement, but initializations of automatic or register variables are ineffective." The phrase "this statement" refers to the compound statement, not to the switch statement; that is, K&R does not say that a switch statement may contain declarations. What it does say is that a switch may contain a compound statement, which, in turn, may contain declarations. You can't have declarations inside a switch unless a compound statement is present.

Space for auto and register definitions is guaranteed to be allocated; however, any initializer list will probably be ignored. Such lists are evaluated at runtime, and since there is no way to execute code in a switch unless it belongs to a case, any such initializations will be ignored (unless you **goto** a label before the first case.) This problem is peculiar to definitions inside a switch since the flow of control passes into a case label rather than just falling into the compound statement.

The key phrase in the draft is in the section "Storage Duration of Objects," where it is stated, "If an initialization is specified, it is performed on each normal entry, but not if the block is entered by a jump to a label." Abnormal entry is using **goto** to jump into a compound statement, or having a switch branch to a case label inside a block. So this whole problem is a property of jumping into compound statements rather than an idiosyncrasy of the switch statement; hence, it is not discussed under the heading "switch" in the Standard.

You could legitimately argue that the Standard should contain more tutorial material to explain these things. And we have added a significant number of footnotes and examples in general. The problem is, it's a Standard definition, not a tutorial, and drawing the boundary involves compromise.

And yes! Your code fragment will work. ♦

Rex Jaeschke is an independent computer consultant, writer, and lecturer. Rex is also a member of the X3J11 C Standards Committee, observer on the ISO C Standards Committee, Founding Editor of The C Journal, *and the C Language Editor of* DEC PROFESSIONAL.

If you would like to comment on or question any aspect of the Proposed ANSI C Standard, write to

Rex Jaeschke
2051 Swans Neck Way
Reston, VA 22091

Rex will endeavor to address your concerns in future columns as space permits.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Heap Expander™ version 2.0

Now your programs can have virtually unlimited heap space using expanded memory, extended memory, disk space, or any combination of the three. And it's all transparent. The Heap Expander's initialization code checks the system's resources and uses whatever is available.

still $59.95*

- Uses LIM-standard expanded memory if present.
- Uses AT-style extended memory if present.
- Swaps data to disk as needed.

Libraries and Source Code for:

- Turbo C
- Microsoft C 4.0 and 5.0
- Turbo Pascal 3.0 and 4.0

Requires an IBM PC, XT, AT, or close compatible with MS-DOS or PC-DOS version 2.0 or above

MC/VISA/COD call
1-800-248-1045 x 100 (US)
1-800-952-5560 x 100 (Idaho)

*Idaho residents add 5% sales tax
Foreign customers add $4.00 for shipping and handling.

The Tool Makers
P.O. Box 8976
Moscow, Idaho 83843
208-883-4979

Number 166 on the Reader Service Card

What have YOU been missing?

C Programmer's Toolbox Volumes I & II

Create better, faster, higher quality and easier to read programs in a fraction of the time. Let your system do the work for you. With 23 powerful, state of the art tools for the IBM PC and compatibles, both beginners and experts will find programming a breeze.

Easy to use. Unlimited program sizes. Online documentation...

CFlow™:	Determine program hierarchy, external R/T library functions, etc.;
CLint™:	More rigorously check program syntax;
CPrint™:	Beautify source programs to user selected formats;
CXref™:	Cross reference and check symbol usage;
CritPath™:	Determine a program's critical path;
PMon™:	Monitor program/OS execution; and more...

At $79.95 per volume or $130 for both with a 30 day money back guarantee, the Toolbox is simply the best value today. The Toolbox works with your existing C compiler(s) and enhances your development environment.

Call and Order Today
Visa, MasterCard Accepted

MMC AD Systems
Box 360845 Milpitas, California 95035
(408) 263-0781
"The C Tool Specialists"

Number 94 on the Reader Service Card

C PROGRAMMER!

10 Important Reasons to Use db_VISTA™ for File Management

1. **db_VISTA is written in C.**
2. **It's fast,** combining B-tree indexing with "network" or direct "set" relationships between records.
3. **It's flexible.** Use **db_VISTA** as a file manager or a complex database; single-user PC to multi-user VAX with millions of records. Upgrade easily with full compatibility!
4. **It's portable.** MS-DOS, UNIX, VMS . . . see list below.
5. **It uses space efficiently.** Non-RAM resident, only *operative functions* are in your run-time program.

And there's more . . .

6. **Royalty-free run-time.**
7. **Complete source code available.**
8. **SQL-based db_QUERY!** Add fast, C-linkable db_QUERY for ad-hoc queries and reports. Build an end-user interface; provide a relational view of the database.
9. **FREE tech support hotline!** 60 days free; for product or application development help. Extended support, consulting, training classes — all available.
10. **30 Day Money-Back Guarantee.** Full refund if not completely satisfied.

That's just the beginning!
Want more details? Call today!

Order Now. It's easy — simply call toll-free. We'll answer your technical questions and get you started . . . fast delivery.

Call Toll-Free Today!
1 (800) db-RAIMA
(that's 1-800-327-2462)
— OR Call 1-206-828-4636

db_VISTA™ Royalty Free Prices

	Object	w/Source
Single-User	$ 195	$ 495
Multi-User	$ 495	$ 990
VAX Multi-User	$ 990	$1,980

db_QUERY prices are the same as above

MasterCard VISA

- **Operating systems:** MS-DOS, UNIX, XENIX, ULTRIX, VMS; more . . .
- **Compilers:** Microsoft, Lattice, IBM, Computer Innovations, Aztec, Turbo C, UNIX, XENIX; more to come . . .

also works with most C libraries!

RAIMA™ CORPORATION
3055 - 112th N.E., Bellevue, WA 98004 USA
(206) 828-4636 Telex: 6503018237 MCI UW

PJ 38

Contact this Advertiser directly

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Converting a Turbo C Program to a TSR

by
Michael J. Young

A memory resident program, or TSR (terminate and stay resident application), is one that permanently establishes itself in memory and then returns control to MS-DOS so that the user may run other programs. The TSR remains dormant in the background until the user presses a designated hotkey, whereupon it springs to life, temporarily suspending the current application and providing instant access to the services it offers.

The module of functions presented in this article allows you to convert a normal Turbo C program into a TSR through a single function call. You may specify the hotkey that is to activate your TSR and the C function in your program that is to receive immediate control when the hotkey is pressed. You can also designate a unique code to prevent the same resident program from being installed more than once. All of the details of installing and activating the program as a TSR are handled invisibly by the function module.

The TSR function module is given in Listing 1, and the header file that you must include in your C program to call the functions is given in Listing 2.

Writing a TSR under MS-DOS is a complex task. The primary challenge is to design a resident program that will peacefully coexist with the foreground program, with any other TSRs that are currently installed, and with the operating system. It is especially important that a TSR written in the C language be able to freely call MS-DOS functions since many of the standard library routines make use of these services. This article presents only one of the many approaches that can be used to achieve these design goals.

This article is divided into two main sections. The first describes how to use the TSR functions to convert a Turbo C program to a memory resident application, and the second explains the inner workings of the functions in the TSR module. Therefore, if you are anxious to begin using the functions to develop a TSR, you need read only the first section. If, however, you are interested in the theory behind the design of resident applications, you will want to read both sections.

The TSR functions furnish an excellent demonstration of the advanced features of Turbo C. Normally it would be quite difficult to write memory resident interrupt handlers in the C language; however, using Turbo C language extensions such as interrupt functions and pseudo variables, it was possible to write the entire module in C, with the exception of two inline assembly language statements.

1. Using the TSR Functions

There are only two functions in the TSR module that may be called directly by your program: **TsrInstall** and **TsrInDos**. This section describes how to call these two functions from your Turbo C program.

```
int TsrInstall (void (*FPtr) (void),int HotKey,VIFP Code)
```

TsrInstall

The function **TsrInstall** has the prototype and performs the following primary tasks:

- It checks to make sure that the program has not already been installed in memory. Each program that is installed by this routine should be identified with a unique value of the parameter **Code**.
- It installs the appropriate interrupt handlers so that when the user presses the shift key combination specified by the **HotKey** parameter, the function given by the **FPtr** parameter receives immediate control.
- It terminates the C program, leaving the entire block of code and data resident in memory.

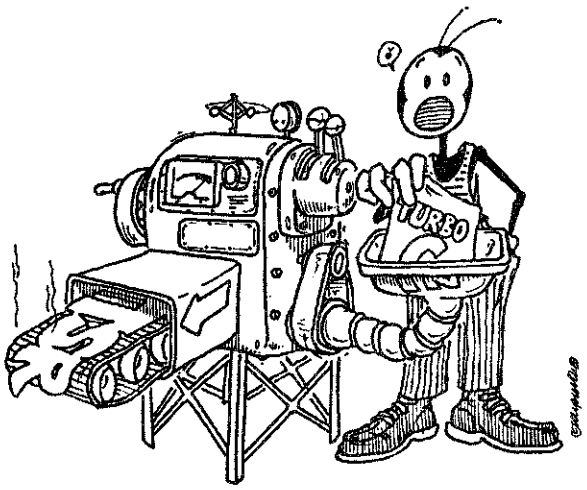
Normally, the **main** function serves only to perform any initializations that are required, to print messages to the user, and to call **TsrInstall**. The parameter **FPtr** should not contain the address of **main**, but rather the address of the actual TSR entry routine: the C function that is to receive initial control when the user presses the hotkey.

The parameter **HotKey** gives the shift key combination used to activate the program. Note that this parameter does not specify either an ASCII code or an extended keyboard code, but rather some combination of shift keys. These combinations may be specified by ORing together the constant identifiers contained in the header file **tsr.h** (Listing 2). For example, the combination of the Alt key and Left Shift key could be specified by the following expression:

```
ALT | LEFTSHIFT
```

The parameter **Code** is of type **VIFP** (void interrupt function pointer, defined in tsr.h). It should be a number between the values 0x00000001 and 0xffffffff that is unique for each distinct program you convert to a TSR. Once a TSR is installed, no other program having the same ID number may

PRODUCED 2005 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



subsequently be installed. By means of this parameter, **TsrInstall** avoids loading redundant copies of the same program. Note that you must use a cast operation to convert a constant number to the appropriate type; for example,

```
(VIFP)0x12345678
```

Since **TsrInstall** terminates the current program, it should obviously never return. If the function does return, an error has occurred and the program has not been installed. In this case, one of the following error codes is passed back to the calling program (these constants are defined in the header file of Listing 2):

- **INSTALLED**: This error code indicates that a TSR bearing the same identification code has already been installed.
- **NOINT**: This value means that there are no free "user" interrupts available. The interrupt vectors in the range from 60h to 67h are designated as available for user programs. The first of these vectors that is free (that is, non-zero) is used to contain the identifying code (the parameter **Code**) assigned to the TSR. If all of these vectors are occupied, **TsrInstall** returns **NOINT**.
- **WRONGDOS**: The TSR functions require DOS version 2.0 or later. If the version is prior to 2.0, this error code is returned.
- **ERROR**: This general code is returned if the installation procedure failed for some unknown reason.

Note that even if the TSR is properly installed, the program may not always receive immediate control when the hotkey is pressed. There are times when it is not safe to interrupt the current process (such as when DOS or the BIOS are engaged in disk activity). The TSR module senses these conditions and simply ignores the hotkey until it is safe to trigger the resident program.

TsrInDos

The function **TsrInDos** has the prototype **int TsrInDos (void)**. This function returns a non-zero value if an MS-DOS service was active when your TSR received control, and it returns 0 if MS-DOS was not active. The purpose of this function is to allow you to determine which DOS functions your program may safely call. Specifically, if **TsrInDos** returns a value of 0, you may directly call any MS-DOS service or call any C library function that invokes MS-DOS (except program termination services). However, if **TsrInDos** returns a non-zero value, *you must not use DOS functions 01h through 0Ch.* This set of forbidden functions manages input and output to the basic character devices (console, printer, and serial port). In Turbo C Version 1.0, the library functions that employ these services are all those defined in the header file **conio.h**, namely: **cgets**, **cputs**, **cprintf**, **cscanf**, **getch**, **getche**, **getpass**, **kbhit**, **putch**, and **ungetch**. In Turbo C Version 1.5, you may safely use the following functions since they no longer employ MS-DOS services: **cprintf**, **cputs**, and **putch**. The reasons that you may not call this set of MS-DOS services are explained in the "The General Strategy" in Section 2.

General Guidelines for Using the TSR Functions

In most respects, you can write a program destined to become a TSR the way you would any other C program. However, you should follow four general guidelines when developing a memory resident program.

Memory Models and the Heap: The first guideline concerns the use of dynamic heap memory in the various Turbo C memory models. If you compile your program under any of the small data models (tiny, small, or medium), you are free to call dynamic memory allocation functions, such as **malloc** and **calloc**. However, if you use a small data model, the minimum size of your TSR in memory is quite large (greater than 64 kilobytes; the example program in Listing 3 occupies approximately 73 kilobytes).

If you compile your program using a large data model (compact, large, or huge), you may not call dynamic memory allocation functions (or use Turbo C library functions that call them). However, under a large data model the minimum size of your TSR in memory is small (the example program in Listing 3 occupies only about 19 kilobytes). These memory model constraints are explained in "Details of the Implementation" in Section 2.

Preserving the Video State: The second guideline is that if your program writes to the screen or alters the cursor, you must carefully save and restore the complete video state of the interrupted program. The functions in the TSR module automatically save and restore almost all of the machine state of the suspended program; the task of preserving the video state, however, is left to the application program. To maintain compatibility with any foreground program, you must save and restore the following elements of the video state:

- the video mode and active display page
- the cursor position and shape
- the screen data (text or graphics)

The example TSR in Listing 3 uses the functions **ScrSave** and **ScrRestore** to preserve the screen data.

Returning: The third guideline is that your TSR must always return from the function that was initially activated (the func-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Programmer's Tools

from The Software Family

Data Base

Btree/ISAM *by Softfocus*: The btree index structure provides rapid access to data based on content with multiple keys per file. Includes full C source code. ISAM automatically keeps data and indices synchronized, reducing your work compared to other btree packages. Multi-user functions available. For use with all popular C compilers. **$99**

QBase *by Crescent Software*: Qbase is a screen designer and relational database for BASIC that lets you quickly create applications including data entry, validation and file management. Qbase data validation is controlled by the database, so all input edits are automatically changed when you modify the database definition. Includes all source code. Specify QuickBasic or TurboBasic when ordering. **$72**

Animation

Grasp *by Paul Mace*: A simple language to create and run animated graphics, demonstrations, and presentations. Images are captured from other applications or created using a built-in paint program. Can be called from other programs and can call other programs. **$69**

Windows and Data Entry

Vitamin C *by Creative Programming* provides a comprehensive set of C functions for windows, menus, data entry & validation. Includes complete C source code. Specify your primary compiler when ordering. **$159**

UI Programmer by Wallsoft: UI is the dBase code generator that produces code in your own personal or company style. UI provides support for all the program features you need like data validation, pop-up boxes, menus and color. Unlike most code generators, however, UI makes it easy for you to design your own templates for code generation. UI improves your dBase programming consistency and productivity without sacrificing control over design and coding style. **$239**

Artificial Intelligence

TransLISP+ *by Coder's Source*: Translisp is intended for real systems development, providing both a comprehensive lisp implementation for AI functions and an interface to C for functions more conveniently handled with a conventional programming language. **Special through 3/31/88: $99 including run-time library.**

Free Catalog & Newsletter

The Software Family carries hundreds of products from dozens of publishers. Call or write for your free copy of our comprehensive catalog and newsletter.

Debugging and Quality Control

Periscope II *by Periscope*: This is the debugger that lets you know where your program is getting hung up, even if you don't know where to set the breakpoint! Just press the NMI hardware button provided with the debugger to interrupt the current program at any time and enter the best debugger available. Don't waste hours guessing where to place you breakpoints or make do with less complete debuggers. For all popular languages and compilers. **$139**

PC Lint *by Gimpel*: Identifies hard-to-find C errors like inconsistent declarations in multi-module programs and "=" instead of "==" in expressions. For all compilers. **$95**

Returns

The Software Family wants you to pay only for the tools you really use. If you order anything featured in this ad (or most of the products in our catalog) you can open and use the product for up to 30 days and still return it if it doesn't meet your needs. Our return rules are simple:

- The item must be shipped to us within 30 days of the day you received it.
- The product must be in salable condition (complete and clean).
- You must prepare a brief letter describing why you were dissatisfied and stating that you have not kept any copies of the product or derived products.
- We will promptly refund your money less shipping charges.

The Software Family

800-443-7176

(415)957-0111

Visa • MasterCard • COD
American Express • Check

- Most orders shipped within 24 hours
- Free UPS shipping on orders over $100 ($3 on orders under $100)
- Federal Express, UPS air, and international service charges: our cost + $3 handling
- $2 COD fee
- No surcharge for credit cards

The Software Family
649 Mission Street, Dept PJ038
San Francisco, CA 94105

Technical Support

The Software Family carefully evaluates features, quality and support before offering a product to our customers. All of the products featured in this ad are backed by excellent technical support from the publisher — in many cases directly from the program developers. All are published by companies with reasonable update policies and with a track record for responding to customer requests for both bug fixes and new features. We at The Software Family further back the products with our own technical support. We are willing and able to intercede with the publisher if your needs aren't satisfied or to research problems caused by the interaction of products we recommend. The We can also provide quotes for custom modification of source-licensed products, in-depth technical consultation or custom-developed products.

©1988 TSF All Rights Reserved

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

tion specified by the **FPtr** parameter). *You must not use the exit function to terminate the program.* The **exit** function is designed to terminate a non-resident program and return to DOS; using it from within a TSR would most likely abort both the TSR and the interrupted foreground program, and probably crash the entire system at the same time.

Critical-Error and Control-Break: The final guideline is that if you are going to use any MS-DOS functions (or call C library functions that invoke DOS services), you should install both a break key handler at the DOS level (interrupt 23h) and a critical error handler (interrupt 24h). The primary reason for installing custom interrupt handlers for break-key and critical error conditions is to prevent the user from simultaneously aborting both the TSR *and* the suspended foreground application through whatever interrupt handlers happen to be installed. (When the TSR is activated, either the standard DOS handlers or those belonging to the foreground application will be installed.) Note that *you must save and restore the original interrupt handlers* (therefore, you cannot simply use the Turbo C functions **ctrlbreak** and **harderr**; the functions **getvect** and **setvect** are useful for obtaining and setting interrupt vectors).

An Example TSR

The program in Listing 3 demonstrates the use of **TsrInstall** and **TsrInDos**. This program consists of two functions: **main** and **Test**. The function **main** calls **TsrInstall** to convert the program to a TSR, specifying **Test** as the function to receive control when the user presses the hotkey. The selected hotkey is Ctrl-Left Shift, and the identifying code is 11111111h. If another TSR is developed that might be installed at the same time as this program, a *different* hotkey combination and identifying code should be chosen. If **TsrInstall** is successful, it will never return; however, if an error occurs in the installation process, control returns to main, which then prints the appropriate error message and terminates with an errorlevel setting of 1.

If **TsrInstall** is successful, control returns to DOS. Subsequently, each time the user presses Ctrl-Left Shift (and it is safe to interrupt the current process), the function **Test** receives control. This function saves and restores the screen contents and displays a popup window. It then uses the function **TsrInDos** to indicate whether DOS is currently active. To prepare the executable file, you must link this program with the TSR module in Listing 1. Note that the program, written to provide a simple example, will work properly only if the system is in an 80 column text mode when interrupted by the TSR; it will fail if either a 40 column or graphics mode happens to be active. Since **Test** does not alter the cursor, it does not need to preserve the cursor parameters, and since it does not call MS-DOS, it does not need to install break-key or critical error handlers.

2. How the TSR Functions Work

The General Strategy

Before we discuss the details of the individual functions in the TSR module, this section provides an overview of the

o new o written specifically for Turbo C o save time developing high performance applications o compatible with the PC, AT, & PS/2 o convert a Turbo C program to a TSR activated by a hotkey o create screens, windows, & menus with an interactive designer o rapidly display screens & strings o read data from specific fields on the screen o display, stack, & move windows o control the cursor, video modes, & display attributes o write data to a large virtual screen o scroll & view a virtual screen through a window o manage the keyboard o manage the printer o swap printer ports o format printed reports & forms o access the DOS print queue from Turbo C o manage directory names, volume labels, & file dates & times o open up to 255 file handles simultaneously o control-break & critical-error handlers o a complete device independent graphics package o an LIM expanded memory interface o over 100 functions o 2 diskette set o ready-to-link library files for all memory models o complete commented source code included o printed manual o many example programs o no royalties o $39.50 introductory price o

Systems Tools for Turbo C

A comprehensive function library by the author of *Systems Programming in Turbo C* (Sybex,1988)

Send check or money order for $39.50 plus $2.50 shipping and handling (California residents please add 6% sales tax). Your Turbo C toolset will be shipped immediately.

Michael J. Young, Software Engineering
P.O. Box 5068
Mill Valley, CA 94942
415/383-5354

Number 168 on the Reader Service Card

For those who need a

FAST C COMPILER with 8087 & software floating point and a full DOS 3 compatible STDIO library
FAST 8086 ASSEMBLER that is stand-alone, or inline with no separate assembly step
FAST LINKER that makes stand-alone .EXE files with both memory and disk-based overlays
SOURCE DEBUGGER for both C and Assembler
FULL SCREEN EDITOR that edits 2 files with no size limits & can invoke the Compiler & Linker
GRAPHICS package with line and text functions

DeSmet DC88

We've combined our popular DeSmet C88 Development Package with the DeSmet D88 Debugger and our Graphics Package. Library source code is available separately.

$99

C Ware

P.O. Box 428, Paso Robles, CA 93447
Phone:(805)239-4620 Telex: 358185 BBS: (805)239-4834
We accept VISA, MC & AMEX. Call now and we'll ship today
945 Spring St #14, Paso Robles, CA 93446

Number 38 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

basic tasks required to install and activate a memory resident program and the general strategies that the TSR module uses to accomplish these tasks.

The most fundamental task required to install a TSR is to load the program into a protected area of memory that will not be overwritten by subsequent applications. Fortunately, MS-DOS provides a service that allows a program to terminate *without* relinquishing the memory it occupies. MS-DOS reserves this area of memory by removing it from its list of free memory blocks and loads all subsequent applications at higher memory addresses. The TSR module accesses this DOS service through the Turbo C library function **keep**. The only complication in using this function is that it is necessary to specify the size of the program that is to be kept resident. The details of using **keep** are discussed in "Details of the Implementation" in this section.

The next task required to create a TSR is to establish some means for activating the program through a hotkey. The basic strategy used by the TSR module is to replace the current keyboard interrupt handler with a Turbo C interrupt function. Each time a key is pressed or released, the interrupt function receives control; this function first calls the former interrupt handler and then simply tests the BIOS shift status flag to see whether the designated hotkey is pressed. If the hotkey is pressed, the routine then goes on to perform other tests and possibly activates the TSR entry function.

Another important duty of the TSR code is to prevent the resident program from being activated at a time when it is not safe to interrupt the current process. The first such dangerous time is when the BIOS is engaged in disk activity. Therefore, the TSR module also installs a handler for interrupt 13h, which is the entry point for the BIOS disk services used by MS-DOS and by some application programs. The Turbo C interrupt 13h function first sets a global flag indicating that BIOS disk activity is in progress, calls the original interrupt 13h routine, and then resets the flag when the original routine returns. Therefore, before the keyboard interrupt handler activates the TSR entry function, it checks this flag and returns immediately if the flag reveals that the disk services are currently in progress.

It is also unsafe to activate a TSR when most MS-DOS services are active. The danger of suspending an MS-DOS service is that the TSR itself may subsequently invoke an MS-DOS function, which would most likely corrupt the stack used by the original invocation of DOS. In general, the MS-DOS kernel cannot be interrupted at an arbitrary point and have its code reused by another process (since the code cannot be recursively re-entered, it is termed nonre-entrant). MS-DOS, however, provides a partial solution to this problem. Whenever DOS is active, it sets an internal flag to a non-zero value; the address of this **indos** flag is returned by the undocumented, but often used, function 34h. Therefore, in addition to checking the BIOS disk activity flag, the keyboard interrupt handler needs to check the **indos** flag and return immediately if either of these flags is non-zero.

This technique for avoiding interrupting MS-DOS, however, has a serious problem. While the command interpreter is waiting for input at the DOS prompt, the **indos** flag is set to 1 since DOS uses one of its own services to read characters. While DOS is waiting at the command prompt, however, it is relatively safe to activate a TSR; in fact, all DOS

LEARN about . . .

Neural Nets

A major upheaval in thinking about artificial intelligence is underway. Interest is accelerating in the "neural net" approach, which is loosely based on human brain research. A tutorial introduction to this exciting new technology is now available in a model called Netwurkz (tm).

Netwurkz eliminates the complicated math found in "unsupervised learning" so that underlying principles may be grasped quickly. The network "learns" when you add new DATA statements (interconnects), or modify existing ones. A trace on/off feature prints the neural net computation in progress. List based internal representation yields speed.

Netwurkz is intended for people not familiar with neural modeling who want to maintain or broaden their technical skills and background. Source for all neural net software and an object-only copy of the PL/D compiler are included

Compilers

PL/D (tm) is a new system language that contains documentation so definitive that your understanding of all languages is enhanced. Like other languages, PL/D is described in narrative form, alphabetically by token and via flow diagrams. However, PL/D also comes with self-compiling source! If you have a question after reading the manual, you can look at the source. If you do not accept the answer, you can change it by modifying the code.

What you learn from PL/D also applies to all compilers: symbol table organization, macro handling, nested structure implementation, cross reference listing, code generation, etc. You will study an unusually efficient implementation: PL/D is 2-3 times faster than the recent "fast" C compilers.

If you are a bright self-sufficient type who wants to master your development tools, PL/D is for you.

PC or compatible with 256K and DOS 2.0 or above and 5.25 inch diskette required.

DAIR Computer Systems 3440 Kenneth Drive, Palo Alto CA 94303 (415) 494-7081

Name: _____
Company: _____
Address: _____
City, State, Zip: _____
Phone: _____
VISA, M/C: _____ Expire: _____

Netwurkz: neural net demo	$79.95	
PL/D: fast self-compiles	$124.95	
Both of above	$154.95	
Sales Tax, CA only		
Ship overseas	$10.00	
TOTAL		

Want to know more? (415) 494-7081

Available Now

Number 5 on the Reader Service Card

Attention Programmers:

Affordable display utilities.

Now you can make screens, demos and graphics quickly and easily. And improve your competitive position at a price you can afford.

Satisfaction guaranteed or your money back!

"Tremendous productivity tools"
*(**PC Week** October 27, 1987)*

"Many applications will benefit from the quick development times and minimal programming required to implement a finished, quality product"
*(**IEEE Software** November, 1987)*

"If you want to include bit-mapped graphics in your application, then this might be the package for you"
*(**Computer Language** June, 1987)*

DisplayExpress™ $69

A fast, efficient and easy-to-use screen manager.

C-Display Librarian™ $69

Professional video, keyboard and graphics functions.

Send check or money order now to:

SYDETECH
SYSTEM DEVELOPMENT TECHNOLOGIES, INC.
43-23 COLDEN STREET, #17C, FLUSHING, N.Y. 11355

For COD orders or the name of the nearest dealer please call (718) 886-0221

Number 74 on the Reader Service Card

PRODUCED 2007 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

★ NOW AVAILABLE FOR C ★ QUICKBASIC ★
★ IBM BASIC/BASICA ★ TURBO PASCAL ★

The Aspen Systems Subroutine Editor (**ASE**) you can call from your programs. Design your own screen layouts - - update in several windows simultaneously. **ASE** is:

- ★ **TRANSPORTABLE** - can be configured for any keyboard and almost any computer or language under MSDOS.
- ★ **VERSATILE** - customize input for any application.
- ★ **COMPLETE** - with full screen edit capabilities and a wide variety of automatic conversions.
- ★ **EASY TO USE** - data and screen layouts described in a single map.
- ★ **FULLY DOCUMENTED** - including examples in BASIC, PASCAL, FORTRAN, C, and even COBOL.
- ★ **AFFORDABLE-**

ASE is still only..... **$99**
DEMO (See what ASE can do for you!)..... **$ 3**

The Aspen Systems Subroutine Package (**ASP**) - the companion package for **ASE** - with many functions unavailable or difficult to perform in high level languages. Like **ASE, ASP** is:

TRANSPORTABLE ★ FULLY DOCUMENTED
EFFICIENT ★ AFFORDABLE

ASP includes 150+ subroutines: conversions; sorts; string, bit, date/time functions; decimal arithmetic, and **MORE**.

ASP is still only..... **$130**

ASE and **ASP** include support and demo programs.

Prices are PPD (continental USA).
Colorado Residents add 3%.

P. O. Box 1163
Grand Junction, CO 81502
(303) 245-3262
VISA/MasterCard accepted

ASPEN SYSTEMS

CP/M and MS-DOS are trademarks of Digital Research and Microsoft, respectively.

Number 86 on the Reader Service Card

Announcing WKS LIBRARY

NEW!

The Lotus "Wrap-Around" for C Programs

Now you can write and read Lotus worksheets directly from a C program!

WKS LIBRARY lets you:

- avoid time-consuming file translation steps
- control the execution of 1-2-3 from inside your application
- use Lotus as a data entry screen in your C program
- generate "live" financial statements with formulas for totals

Feature this:

- reads and writes .WKS, .WK1 and .WR1 worksheets
- writes integers, floats, strings, formulas, macros and dates using *uprintf()*
- reads using *uscanf()*, converting column contents to C variables according to format specs
- has low-level functions for manipulating individual cells
- provides Lotus control functions such as: formats, column widths, initial cursor position, range names, cell protection
- supports Lattice, Microsoft & Turbo compilers for DOS, plus most UNIX environments

Source Code provided **No Royalties on executable programs**

Only $89
(includes free 800-line support)

VISA

ORDER TODAY!
Toll-free
(800) 367-9882

Tenon Software, Inc.
1980 - 112th NE, #250, Bellevue, WA 98004
(206) 453-1914 (in Washington state)

Number 163 on the Reader Service Card

"If you need source code, make sure your wallet is wide open or get VITAMIN C."
- Computer Language, June, 1987

Vitamin C

It's good for your system!

- Professional C function library
- 30 day money back guarantee
- Multiple bullet proof windows
- Easy full screen data entry
- Unlimited data validation
- Context sensitive help manager
- Menus like Lotus and Mac
- Programmable keyboard handler
- Text editor routines
- No royalties or runtime fees
- Library source included FREE
- Free technical support
- Free BBS at (214)418-0059
- Supports all major compilers including Microsoft 5.0
- VCScreen code generator too!
- UNIX version avaialable

Fast, flexible, versatile, reliable. Vitamin C delivers the vital combination that software professionals demand. Highly efficient, professionally crafted C code provides lightning fast displays required by today's window intensive programs.

High level functions provide maximum productivity and require little supporting code. Plus, Vitamin C's versatile, open ended design is full of hooks so you can intercept and plug-in special handlers to customize or add features to most routines.

VCScreen, our screen painter / code generator speeds your development even more! Simply draw your input forms using our interactive design editor and generate perfect C source code ready to compile and link with the Vitamin C library.

Vitamin C $225
Includes all source code FREE! For IBM PC, XT, AT, PS/2 and true compatibles. Specify compiler when ordering.

VCScreen $99.95
For IBM PC, XT, AT, PS/2 and true compatibles. Requires Vitamin C.

ORDER NOW!

We ship UPS. Please include $3 for ground, $6 for 2-day air, $20 for overnight, or $30 if outside the U.S. Texas residents add 7 1/4 % sales tax. All funds MUST be in U.S. dollars drawn on a U.S. bank. Visa & MasterCard accepted.

(214)416-6447

creative PROGRAMMING
Box 112097 • Carrollton, Texas 75011

Windows • Data Entry • Menus • Help • Text Editing
Plus... All Source Code FREE!

Contact this Advertiser directly

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

function calls can be used at this time except those in the range 01h-0Ch. Therefore, the TSR needs some way of knowing that although the **indos** flag is not 0, DOS is simply waiting at the command line and the TSR function may be safely called. Again, MS-DOS provides a solution: While DOS waits for user input at the prompt, it continually invokes interrupt 28h (which is used to activate background processes such as print spooling and is known as the DOS *idle interrupt*). Therefore, the occurrence of interrupt 28h is a *sufficient indication* (regardless of the state of the **indos** flag) that the TSR may be activated. Accordingly, the TSR module installs a third and final interrupt handler, which intercepts interrupt 28h and provides a second door to the TSR.

Therefore, there are two parallel mechanisms for activating the TSR when the hotkey is pressed. First, the TSR may be activated through the keyboard interrupt handler, which must first check both the BIOS flag *and* the **indos** flag. Second, it may be activated through the interrupt 28h handler, which needs to check only the BIOS flag. The TSR knows that it was activated through interrupt 28h if the function **TsrInDos** returns true; in this case, as mentioned in the previous section, it must not call DOS functions 01h through 0Ch since

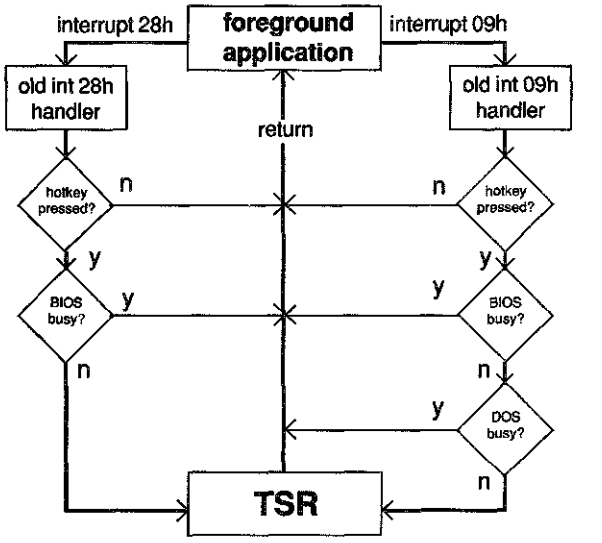


Figure 1. The parallel mechanism for activating a TSR

these functions use the same stack that DOS was using when it was interrupted. This parallel mechanism for activating a TSR is illustrated in Figure 1.

Once it has been determined that it is safe to activate the TSR, the final major task is to save the complete machine state of the interrupted program, set up the runtime environment for Turbo C, and call the TSR entry function that was specified at installation time. When the entry function returns, the machine state must be restored before returning to the suspended application.

Details of the Implementation

The installation function **TsrInstall** begins by testing to see if the TSR has already been installed in memory, by searching through the user interrupts (numbers 60h ot 67h) for the identification number passed in the parameter **Code**. If the code is not found, indicating that the program has not already been installed, the function proceeds to place the code value in the first unused (that is, equal to 0) interrupt vector within this range. **TsrInstall** next checks that it is running under DOS version 2.0 or later since prior versions do not support some of the functions required to install a memory resident program.

Next, **TsrInstall** saves the following important data items that will be required when the TSR is activated:

- The Turbo C stack segment and stack pointer are saved so that the TSR can later use the C stack rather than borrow the stack belonging to the interrupted program. The stack segment value is obtained directly from the corresponding pseudo variable; the stack pointer value, however, is obtained from the **bp** register, which contains an address closer to the end of the stack segment than the current value of **sp** (even a few more bytes could be squeezed out of the stack by calculating the original value of the **sp** register when the program began executing).
- The hotkey shift mask is saved in the variable **HotKeyMask**.
- The address of the TSR entry routine, given by the parameter **FPtr**, is stored in the function pointer **UserRtn**.
- The segment and offset values of the current disk transfer address (DTA) are saved so that the C DTA can be restored before the TSR receives control.
- The address of the **indos** flag, discussed in Section 1 under the subheading "TsrInDos," is obtained through DOS service 34h and is stored in the far pointer **InDosPtr**.
- Finally, **TsrInstall** saves the current interrupt vectors for the three interrupt handlers that will be installed.

Once the required data have been stored, **TsrInstall** installs handlers for interrupts 28h (the DOS "idle" interrupt), 13h (the BIOS disk services), and 09h (the hardware keyboard interrupt). The last task performed by **TsrInstall** is to calculate the size of the program and to call **keep** to terminate the program while leaving the code and data resident in memory. The first parameter passed to **keep** sets the DOS **errorlevel** flag to 0 to indicate a normal program termination. The second parameter specifies the size of the block of memory to keep resident in memory. The size must be given in 16-byte paragraphs and is obtained by subtracting two predefined Turbo C variables, **_heapbase** and **_psp**. The variable **_heapbase** is a far pointer to the first byte of memory beyond the end of the stack segment and marks the end of memory that normally remains allocated to a Turbo C program. The segment value of this far pointer is extracted using the macro **FP_SEG**. The predefined variable **_psp** contains the segment address of the *program segment prefix* (PSP), which is the 256-byte record containing system information that is always located at the beginning of the program in memory. Thus, the total number of paragraphs allocated to the program may be obtained by subtracting these two segment addresses.

Figure 2 illustrates the basic layout of a Turbo C program in memory for small data and large data memory models. This figure shows the positions of the variables **_psp** and **_heapbase** and also reveals the basis for the memory model constraints discussed in Section 1's "General Guidelines" under "Memory Models and the Heap." In the *small* data models, an entire 64 kilobyte block is arbitrarily allocated for the data, heap, and stack. Thus, the memory allocation for a small TSR may be considerably larger than necessary. However, since

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

the heap is located safely within the allocated block, heap memory functions such as **malloc** can be used with impunity.

In the *large* data models, the data and stack segments are

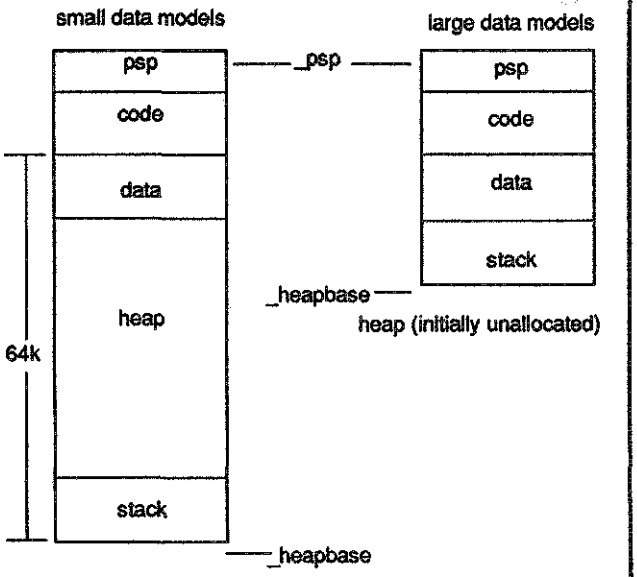


Figure 2. Memory plan of Turbo C programs

only as large as necessary, and the heap is *outside* the allocated memory (with a transient program, heap memory can be dynamically requested from DOS as it is required; however, in a TSR this area would be occupied by another program and would be unavailable). Thus, a small TSR will occupy only the memory actually required for the code and data. However, heap allocation functions will fail since the heap area is not available. Theoretically, it should be possible to avoid this problem by retaining memory beyond **_heapbase**; for example the following command should leave room for a 1000 paragraph (16000 byte) heap:

```
keep (0, FP_SEG (_heapbase) - _psp + 1000);
```

However, for some reason, **malloc** fails to use this additional memory correctly (examination of the library source code may reveal the reason and suggest a solution).

The function **NewInt13** is the interrupt 13h handler. This function simply increments a flag indicating BIOS disk activity (**InBios**), calls the original interrupt handler, and then resets the **InBios** flag. Note that since the BIOS services return information through the **AX** register and the flags, these two values must be assigned to the appropriate stack locations through the corresponding parameters, **AX** and **FLAGS** (otherwise, when the Turbo C interrupt function restores the old register values at function termination, the new register values assigned by the BIOS would be lost). Unfortunately, it is necessary to use a few inline assembler instructions since there is no pseudo variable for the **FLAGS** register.

The function **NewInt09** is the keyboard interrupt handler. This function first calls the former interrupt handler through the function pointer **OldInt09**. It then disables the interrupts so that it will not be interrupted in the middle of testing and setting the **Busy** flag. This flag serves to prevent the TSR from being recursively re-entered while it is active. (Note that

DEVELOPMENT TOOLS FOR C

- A COLLECTION OF "TOOL BOXES" FOR THE DEVELOPMENT OF POWERFUL C PROGRAMS
- DESIGNED TO BE USED BY BOTH NEW AND EXPERIENCED C PROGRAMMERS
- EACH TOOL BOX IS FULLY DOCUMENTED -- WITH EXAMPLES

SYSTEM TOOL BOX
- ACCESS THE POWER OF YOUR PC
- KEYBOARD & SCREEN MANAGEMENT
- GRAPHICS & MUCH MORE
- 184 PAGE USER'S GUIDE

LIB + SRC $35

UTILITY TOOL BOX
- GENERAL PURPOSE FILTERS
- UTILITY PROGRAMS
- GENERAL PURPOSE FUNCTIONS
- 86 PAGE USER'S GUIDE

LIB + SRC $25

WINDOW TOOL BOX
- COMPLETE WINDOW CONTROL
- BUILD WINDOW DATABASES
- FULL ERROR REPORTING
- 84 PAGE USER'S GUIDE

LIB $45 -- SRC $125

MENU TOOL BOX
- VERTICAL/HORIZONTAL MENUS
- CREATE MENU DATABASES
- FULL ERROR REPORTING
- 78 PAGE USER'S GUIDE

LIB $35 -- SRC $75

- HARDWARE -- IBM PC, TI PRO OR TANDY 1000
- COMPILER -- MICROSOFT™ C (3.00 & LATER)

- CALL OR WRITE FOR FREE INFO & DEMO PROGRAM
- TECHNICAL SUPPORT IS UNLIMITED & FREE
- NO ROYALTIES

FLEMING SOFTWARE
5741 Wagonwheel Trail
Castle Rock, CO 80104
(303) 688 - 3584

Number 118 on the Reader Service Card

WINDOWS – MENUS – DATA ENTRY – SCREENS

HI-SCREEN XL™

If you program, you must try HI-SCREEN XL !

HI-SCREEN XL is the only tool that gives you all of the following (and much more) whatever programming language you use:

- Unlimited number of screens & windows
- Overlapping windows for menus, data entry, help screens
- Pull-down, pop-up and Lotus-style menus
- On-line help management
- Field checking for type, range, format,...)
- Field by field and/or full screen input modes
- Scrolling capability
- Full-featured WYSIWYG screen editor
- Redefinable function keys
- Data entry test mode under the editor
- True ease of use, and
- Ease of maintenance: modify screens without recompiling your program.

Includes the HI-SCREEN XL Toolbox:
- Capture existing screens from other applications
- Trace compiled programs
- Print screens with clear data field description
- Use screens/windows/menus from DOS and Batch files
- Create titles & logos with predesigned font screens
- ...

Softway, Inc.
PC/Soft Product Line
500 Sutter St., Suite 222
San Francisco, CA 94102

$149 - Risk Free
Visa, M/C welcome
Call (800) 338-2652
in CA (415) 397-4666

For PASCAL, C, dBASE, BASIC, COBOL, FORTRAN

Number 110 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

since the TSR switches to a fixed location in a local stack and uses static and external variables to store data, the code is *not* re-entrant.) **NewInt09** next calls **GetShift** to see if the hotkey is pressed. If the hotkey is currently pressed, the function tests whether the BIOS disk services are active through the **InBios** flag and whether DOS is busy though the far pointer to the **indos** flag, **InDosPtr**. If all tests are passed, the function **Activate** is called, which performs all the steps necessary to activate the TSR entry function. Upon return from **Activate**, the **Busy** flag is switched off and the interrupt handler returns.

The function **NewInt28** is the interrupt 28h handler and performs the same tasks as **NewInt09** except that it does *not* check whether MS-DOS is active, for reasons described earlier in this section under "The General Strategy."

The function **Activate** can be called by either **NewInt09** or **NewInt28** when one of these interrupt handlers has detected that the hotkey is pressed and that it is safe to invoke the TSR. **Activate** first switches from the stack belonging to whatever program was interrupted to the Turbo C stack. There are two important reasons for switching to the C stack:

- First, the current stack may be quite small. Switching to the C stack provides the TSR with the full stack space normally allocated to a Turbo C program.
- Second, under the small data models, the C compiler assumes that the stack segment is equal to the data segment. Specifically, when an address is passed to a function, only the offset portion is used. The offset, however, may be relative to either the data segment (for external and static variables) or the stack segment (for automatic variables). The function simply uses the offset in conjunction with the data segment register, assuming that this register is *equal* to the stack segment register. If the TSR did not switch to the C stack, this assumption would be false and C functions would be unable to access automatic data through addresses passed as parameters.

Activate next saves the old disk transfer address and switches to the Turbo C disk transfer address that was saved by the installation procedure. Next, before calling the TSR routine, **Activate** informs MS-DOS that a new process is active by setting DOS's record of the program segment prefix to the correct value for the C program. DOS identifies a process by the segment address of its program segment prefix, and there are two undocumented functions for manipulating DOS's internal record of the current PSP. Function 51h *gets* the current program segment prefix, and function 50h **sets** it. The old PSP address is stored in the variable **OldPsp**, and the new address is obtained from the predefined Turbo C variable **_psp**. An advantage of switching to the C PSP is that MS-DOS stores file handles within the PSP and there may be insufficient room to open new file handles in the existing PSP (for example, if the TSR interrupts a data base application that uses all 20 available handles). Note, however, that this DOS function can be called only if DOS is not currently active, and therefore the PSP is switched only if the **indos** flag is 0.

Finally, the TSR entry function is called through the function pointer **UserRtn** that was defined in the installation procedure. Upon return from the TSR routine, **Activate** restores all the values that were saved before calling the routine: the PSP, the disk transfer address, and the stack. At this point **Activate** returns to the interrupt handler that called it (**NewInt9** or **NewInt28**).

The function **TsrInDos** simply returns the current value of the **indos** flag, which is obtained through the pointer **InDosPtr**. ♦

Note: This article was adapted from a chapter in the author's book *Systems Programming in Turbo C* (Sybex, 1988). The listings that accompany the article are available on the *PJ* Issue Disk for Volume 6.2 and on the *PJ* bulletin board services. See the Housekeeping page. Additional source code for the TSR functions is available from the author as part of the Systems Tools for Turbo C function library.

Michael J. Young is a software engineering consultant and developer of software tools for C programmers. He is a member of the ANSI committee on the standardization of the C language and the author of several books on C programming under MS-DOS. Michael can be contacted at

P.O. Box 5068
Mill Valley, CA 94942
(415) 383-5354

Code follows.

Form-Fill Toolkit

Product: C Library
Price: $59.95
Features:

Cursor Control
Sound & Printer Control
Fast Screen Access
Keyboard Access
Edited Field Entry
Pop-Up Window Support
Display Text Windows
Menu & List Selections
Form-fill Routines

Compiler Support
Microsoft 4.0 : Y
Microsoft 5.0 : Y
Quick C : Y
Turbo C : Y
Other:

↑↓ Scroll F1 Help F2 Save F4 Cancel Esc No Save

Provide a consistent user interface for your programs with over 100 documented functions.
SPECIAL! Includes powerful multi-record data collection module with fully commented pristine source code.

Anji Software, Inc.
P.O. Box 297
Cedar Grove, NJ 07009
(201) 239-4328

• Library Source Code - $119.95
• Specify compiler when ordering
• No Royalties required
• Demo Diskette $10.00

We accept Visa and M/C
NJ Residents add Sales Tax
Shipping $3.00 Domestic
$10.00 Foreign

Number 162 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Code From "Turbo C TSR" by Michael J. Young — 1

Listing 1

```
#pragma inline

#include <dos.h>
#include "tsr.h"

static unsigned int TCSS;                       /* Turbo C stack segment.       */
static unsigned int TCSP;                       /* Turbo C stack pointer.       */
static void (*UserRtn) (void);      /* Pointer to user's start routine.     */
static int HotKeyMask;              /* Hot key shift mask.                  */
static unsigned int TCDtaSeg;       /* Turbo C Disk Transfer Area segment.  */
static unsigned int TCDtaOff;       /* Turbo C Disk Transfer Area offset.   */
static char far *InDosPtr;          /* Pointer to DOS 'indos' flag.         */
static void interrupt (*OldInt28) (void);     /* Old int 28h vector.        */
static void interrupt (*OldInt13) (void);     /* Old int 13h vector.        */
static void interrupt (*OldInt9) (void);      /* Old int 09h vector.        */
static void interrupt NewInt28 (void);        /* New int 28h handler.       */
                                              /* New int 13h handler:       */
static void interrupt NewInt13  (unsigned BP, unsigned DI, unsigned SI,
                                 unsigned DS, unsigned ES, unsigned DX,
                                 unsigned CX, unsigned BX, unsigned AX,
                                 unsigned IP, unsigned CS, unsigned FLAGS);
static void interrupt NewInt9 (void);         /* New int 09h handler.       */
static int GetShift (void);         /* Returns BIOS shift status flag.      */
static int Busy = 0;                /* Flag indicates TSR is active.        */
static void Activate (void);        /* Routine to activate user's TSR.      */
static int InBios = 0;              /* Flag indicating BIOS disk activity.  */
extern void far *_heapbase;         /* Defined by Turbo C;  holds break addr.*/

int TsrInstall (void (*FPtr) (void),int HotKey,VIFP Code)
/*
     This routine terminates the C program, but leaves the code resident in
     memory.  It installs interrupt handlers so that when the shift key
     combination specified by 'HotKey' is pressed, the function pointed to
     by 'FPtr' receives control, provided that conditions are safe.
*/
     {
     unsigned int i;
     int Found = 0;

     for (i=0x60;i<=0x67;++i)           /* Search 'user' interrupt vectors  */
          if (getvect (i) == Code)      /* to see if Code is present        */
               {                        /* indicating that the TSR is       */
               Found = 1;               /* already installed.               */
               break;
               }
     if (Found)                         /* Already installed, therefore     */
          return (INSTALLED);           /* return error message.            */
     for (i=0x60;i<=0x67;++i)           /* Not already installed, therefore */
          if (getvect (i) == (VIFP)0)   /* search for a free (==0) user     */
               {                        /* interrupt vector.                */
               setvect (i,Code);        /* Found a free vector;  therefore  */
               Found = 1;               /* place special code in vector.    */
               break;
               }
     if (!Found)                        /* No free vectors;  therefore      */
          return (NOINT);               /* return error code.               */

     if (_osmajor < 2)                  /* Test that OS version is >= 2.0.  */
          return (WRONGDOS);            /* Wrong DOS version.               */

     TCSS = _SS;                        /* Save Turbo C stack segment.      */
     TCSP = _BP;                        /* Save Turbo C stack pointer.      */
     HotKeyMask = HotKey;               /* Save hotkey shift mask.          */
     UserRtn = FPtr;     /* Save address of the routine that is activated   */
                         /* when the hotkey is pressed.                     */
     _AH = 0x2f;                        /* Save old Disk Transfer Address.  */
     geninterrupt (0x21);               /* Invoke MS-DOS function to get    */
     TCDtaSeg = _ES;                    /* current Disk Transfer Address.   */
     TCDtaOff = _BX;

     _AH = 0x34;                        /* Get pointer to 'indos' flag using*/
     geninterrupt (0x21);               /* the undocumented function 34h.   */
     InDosPtr = MK_FP (_ES,_BX);

     OldInt28 = getvect (0x28);         /* Save old interrupt vectors.      */
     OldInt13 = getvect (0x13);
     OldInt9 = getvect (0x9);

     setvect (0x28,NewInt28);           /* Initialize new interrupt vectors.*/
     setvect (0x13,NewInt13);
     setvect (0x9,NewInt9);

     keep (0, FP_SEG (_heapbase)-_psp); /* Terminate and stay resident.     */

     return (ERROR);                    /* 'keep' should not return,        */
                                        /* therefore something has gone     */
                                        /* wrong.  Return general error code*/
     } /* end TsrInstall */


static void interrupt NewInt28 (void)
/*
     This is the new interrupt 28, the 'DOS idle interrupt', handler.
*/
     {
     (*OldInt28) ();            /* Chain to previous interrupt 28 handler.  */
     disable ();                /* Disable interrupts to test and set       */
     if (Busy)                  /* 'Busy' semaphore.                        */
          return;
     Busy = 1;
     enable ();                 /* Re-enable interrupts.                    */
     if ((GetShift () & HotKeyMask) != HotKeyMask)     /* Test if hot key   */
          {                                            /* is pressed.       */
          Busy = 0;
```

Code Continues. . .

Locate C Bugs before they Bite with

PC-lint

PC-lint will analyze your C programs (one or many modules) and uncover glitches, bugs, quirks, and inconsistencies. It will catch subtle errors before they catch you. By examining multiple modules, PC-lint enjoys a perspective your compiler does not have.

"... a remarkable well thought-out product which will check for just about every conceivable coding error ... Its value increases with frequent use ... we confidently recommend PC-lint."

Andrew Binstock, The C Gazette

"PC-lint has everything going for it: flexibility, speed, good documentation, and a reasonable price. I exercised the product daily on a large, working, project to see if I could include it in my development tools. The answer is a definite YES."

Stephen D. Cooper, Blue Notes
San Francisco PC Users Group

"... friendliness is a prime consideration in PC-lint. Gimpel has provided ways to make Lint shut up about all those errors you either know or don't care about. If Unix-Lint was implemented as well, I would use it more."

Don Malpass, IEEE Software

Gimpel Software

3207 Hogarth Lane
Collegeville PA 19426
(215)584-4261

PRICE: $139.00 first copy, $100 each additional, MC, VISA, COD, PA residents add 6% sales tax, Outside USA add $15.

Runs on MS-DOS, works with any C compiler - direct support for 12 major C compilers including Microsoft 5.0, Turbo, C86+, Lattice, Datalight, Desmet

PC-lint is a trademark of Gimpel Software.

Please contact this advertiser directly

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

NROFF/PC™

The **REAL** *Thing for* **DOS**

NROFF/PC is a complete text formatting system for *MS-DOS* systems. Including:

NROFF The *powerful* UNIX text formatter

TBL A tool to assist with the layout of tabular material in *Nroff* documents

MM A comprehensive *Nroff* macro package for preparing books and technical manuals

NEQN A tool for describing mathematical equations in *Nroff* documents

- All tools are a complete port from the AT&T Documentor's Workbench 2.0
- It's **Fast!** We've modified *Nroff* especially for DOS for lightning speed
- Supports any Dot Matrix printer and many laser printers
- Specially Priced At **$99**
- A complete *Troff* typesetting system is available **NOW** for LaserJet and PostScript printers on **MS-DOS** for *$695*, XENIX and Microport UNIX for *$795*.

Elan

Elan Computer Group, Inc.
410 Cambridge Ave., Suite A
Palo Alto, CA 94306
(415) 322-2450

Visa and MasterCard Accepted

Unix is a tradmark of AT&T
MS-DOS and Xenix are tradmarks of Microsoft

Number **173** on the Reader Service Card


```
          return;
          }
     if (InBios)              /* Test if BIOS disk services is active.      */
          {
          Busy = 0;
          return;
          }
     Activate ();             /* Conditions are safe, therefore activate TSR*/
     Busy = 0;                /* Reset the active semaphore.                */
     return;

     } /* end NewInt28 */


static void interrupt NewInt13 ( unsigned BP, unsigned DI, unsigned SI,
                                 unsigned DS, unsigned ES, unsigned DX,
                                 unsigned CX, unsigned BX, unsigned AX,
                                 unsigned IP, unsigned CS, unsigned FLAGS)
/*
     This is the new interrupt 13, BIOS disk services, handler.  This
     function serves only to set a flag while BIOS interrupt 13h is active.
*/
     {
     ++InBios;                /* Set flag.                                  */
     (*OldInt13) ();          /* Chain to old interrupt 13h handler.        */
     --InBios;                /* Reset flag.                                */
     AX = _AX;                /* Must pass on the information returned by   */
     asm pushf                /* the BIOS in AX and flag registers.         */
     asm pop FLAGS

     } /* end NewInt13 */


static void interrupt NewInt9 (void)
/*
     This is the new handler for interrupt 09, the hardware interrupt
     activated by the keyboard.
*/
     {
     (*OldInt9) ();       /* Chain to prior interrupt 09 handler.           */
     disable ();          /* Disable interrupts to test and set 'Busy'      */
     if (Busy)            /* semaphore.                                     */
          return;
     Busy = 1;
     enable ();           /* Enable interrupts.                             */
     if ((GetShift () & HotKeyMask) != HotKeyMask)     /* Test if hot key   */
          {                                            /* is pressed.       */
          Busy = 0;
          return;
          }
     if (InBios || *InDosPtr)      /* Test if EITHER DOS or the BIOS disk   */
          {                        /* services are active.                  */
          Busy = 0;
          return;
          }
     Activate ();             /* Conditions are safe, therefore activate TSR*/
     Busy = 0;                /* Reset the active semaphore.                */
     return;

     } /* end NewInt9 */


static unsigned int OldSS;
static unsigned int OldSP;
static unsigned int OldPsp;
static unsigned int OldDtaSeg;
static unsigned int OldDtaOff;

static void Activate (void)
/*
     This function is called either by 'NewInt9' or by 'NewInt28', when
     conditions are safe, to activate the user's TSR entry routine.
*/
     {
     register int DSSave;     /* Temporary storage for C data segment.      */

     OldSS = _SS;             /* Switch to Turbo C's own stack.             */
     OldSP = _SP;
     _SS = TCSS;
     _SP = TCSP;

     _AH = 0x2f;                         /* Save old Disk Transfer Address.  */
     geninterrupt (0x21);
     OldDtaSeg = _ES;
     OldDtaOff = _BX;

     _DX = TCDtaOff;                     /* Set Turbo C Disk Transfer Address*/
     DSSave = _DS;
     _DS = TCDtaSeg;
     _AH = 0x1a;                         /* DOS set DTA service.             */
     geninterrupt (0x21);
     _DS = DSSave;

     if (!*InDosPtr)                     /* Save the old PSP and set the PSP */
          {                              /* to the Turbo C value, using the  */
          _AH = 0x51;                    /* undocumented functions 51h and   */
          geninterrupt (0x21);           /* 50h.  Note that this procedure   */
          OldPsp = _BX;                  /* is safe only if DOS is not       */
          _BX = _psp;                    /* active.                          */
          _AH = 0x50;
          geninterrupt (0x21);
          }

     (*UserRtn) ();                      /* Call the user's Turbo C function.*/
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
     if (!*InDosPtr)                    /* Restore the PSP, if DOS is not   */
          {                             /* active.                          */
          _BX = OldPsp;
          _AH = 0x50;
          geninterrupt (0x21);
          }

     _DX = OldDtaOff;                   /* Restore Old Disk Transfer Address*/
     DSSave = _DS;
     _DS = OldDtaSeg;
     _AH = 0x1a;
     geninterrupt (0x21);
     _DS = DSSave;

     _SS = OldSS;                       /* Restore the old stack.           */
     _SP = OldSP;
     return;

     } /* end Activate */


int TsrInDos ()
/*
     This function returns zero if DOS is not currently active, and a
     non-zero value otherwise.
*/
     {
     return (*InDosPtr);       /* Uses the 'indos' flag maintained by DOS.   */

     } /* end TsrInDos */


int GetShift ()
/*
     This function returns the BIOS shift status word.
*/
     {
     return *((int far *)0x00400017);

     } /* end GetShift */
```

Listing 2

```
/* Error codes returned by 'TsrInstall' if it cannot install the TSR.       */

#define ERROR       1        /* General, undefined error.                   */
#define WRONGDOS    2        /* Version of DOS prior to 2.0.                */
#define INSTALLED   3        /* TSR has already been installed in memory.   */
#define NOINT       4        /* No free user interrupts are available.      */

/* Constant definitions for hotkey shift mask.                              */

#define RIGHTSHIFT       0x0001
#define LEFTSHIFT        0x0002
#define CONTROL          0x0004
#define ALT              0x0008
#define SCROLLSTATE      0x0010
#define NUMLOCKSTATE     0x0020
#define CAPLOCKSTATE     0x0040
#define INSERTSTATE      0x0080
#define CTRLNUMLKSTATE   0x0800
#define SCROLLLOCK       0x1000
#define NUMLOCK          0x2000
#define CAPSLOCK         0x4000
#define INSERT           0x8000

/* Variables and functions.                                                 */

typedef void interrupt (*VIFP) ();      /* Void Interrupt Function Pointer. */
int TsrInstall (void (*FPtr) (void),int HotKey, VIFP Code);
int TsrInDos (void);
```

Listing 3

```
#include <stdio.h>
#include <process.h>
#include <conio.h>
#include <dos.h>
#include <mem.h>
#include <bios.h>
#include "tsr.h"

static void Test (void);                /* The TSR demo routine.            */
static void ScrSave (void);             /* Screen functions.                */
static void ScrRestore (void);
void ScrPutS (char *String,int Attr,int Row,int Col);

main ()
     {
     /* Install 'Test' as TSR function;  Ctrl-Left Shift is hotkey.         */
     switch (TsrInstall (Test,CONTROL | LEFTSHIFT,(VIFP)0x11111111))
          {
          case WRONGDOS:
               printf ("Cannot install TSR;  must have DOS version 2.0 "
                       "or higher.\n");
               exit (1);
          case INSTALLED:
               printf ("TSR already installed.\n");
               exit (1);
          case NOINT:
               printf ("Cannot install;  no free user interrupts.\n");
               exit (1);
          case ERROR:                   /* General error case.              */
          default:
               printf ("Error installing TSR function.\n");
               exit (1);
          }
```

Code Continues...

vLIB

the user interface library for C

A comprehensive, easy to use library of custom C functions for building sophisticated PC applications.

Windows • *overlapping, tiled, built-in window manager*

Menus • *vertical, horizontal ("Lotus" style), pulldown ("Mac" style)*

Forms • *single call input and display, multiple field types*

Pop Ups • *messages, prompts, selection lists*

Full Editing Input

High Speed Display

... and much, much more!

Give your programs a professional look! Develop applications faster!

Compilers supported: Microsoft C and Quick C, Borland Turbo C, Lattice C, and Datalight C.

All memory models supported.

Demo disk with usable sample programs, source, and reference card . . . $10

Library with reference card and detailed programmer's manual $99

Library (as above) with full source code $149

Prices include shipping, CA residents add 7% sales tax.

No royalties. Site license available.

Call or write today:

Pathfinder Associates
291 Madrone Avenue
Santa Clara, CA 95051
(408) 984-2256

Visa and MasterCard accepted.

Number 145 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Q. How many programmers does it take to maintain a MAKE dependency file?

A. NONE! *If you use* VersiMAKE™

VersiMAKE™ is a full-featured MAKE utility that includes:

- **Dependency Generation**
  Derives your system's dependencies, through analysis of its C and MASM source files. Say goodbye to manual maintenance of MAKE dependency files!
- **Wild Card File Name Matching**
  Analyzes an entire collection of source files with a single statement.
- **Nested Include File**
  Handles standard C and MASM "include" conventions, and the INCLUDE environment variable.
- **Powerful Macro Facilities**
  Built-in macros, user-defined macros, and environment variables.
- **Analytical Reports**
  Shows the entire Include file hierarchy for each source file analyzed, and all of the parent source files for each Include file.

Q. How many programmers does it take to trace a symbol thru your system?

A. ONE! *If you use* VersiCREF™

VersiCREF™ is a unique utility that creates a Master Cross-Reference of your entire system.

- **Multi-Lingual**
  Handles C, assembler, or both.
- **Flexible**
  File names with line numbers, or file names alone. Global and local symbols, or globals alone.
- **Powerful**
  Easily handles systems containing 100 source files or more.

VersiMAKE™	$125
VersiCREF™	$75
Both	$150

(Outside U.S., add $5 for shipping and handling)

800-334-4096
(In NJ, 609-871-0202)
MC/VISA/AMEX accepted

SUMMIT INFORMATION SYSTEMS, INC.
73 East Lane, Willingboro, NJ 08046

Number **158** on the Reader Service Card

Young Code, Continued


```
     } /* end main */


static void Test (void)
/*
     This routine is called when the hotkey is pressed.  It displays a
     window and indicates whether DOS is currently active.
*/
     {
     int Row = 6;
     int Col = 24;

     ScrSave ();
     ScrPutS ("+--------------------------------+",0x0f,Row++,Col);
     ScrPutS ("|                                |",0x0f,Row++,Col);
     ScrPutS ("|        T S R   D E M O         |",0x0f,Row++,Col);
     ScrPutS ("|                                |",0x0f,Row++,Col);
     ScrPutS ("|   press any key to continue    |",0x0f,Row++,Col);
     ScrPutS ("|                                |",0x0f,Row++,Col);
     ScrPutS ("+--------------------------------+",0x0f,Row++,Col);

     if (TsrInDos ())
          ScrPutS ("Now in DOS",0x0f,9,35);
     else
          ScrPutS ("NOT in DOS",0x0f,9,35);
     bioskey (0);
     ScrRestore ();

     } /* end Test */

/*** screen functions ******************************************************/

static char SaveScr [4000];

static void ScrSave (void)
     {
/*
     This function saves the current contents of the screen in a buffer.
*/
     int VideoSeg;

     if (*(char far *)0x00400049 == 7)       /* Test video mode.            */
          VideoSeg = 0xb000;                 /* Monochrome video memory.    */
     else
          VideoSeg = 0xb800;                 /* Color video memory.         */
     movedata (VideoSeg,0,FP_SEG ((char far *)SaveScr),
               FP_OFF ((char far *)SaveScr),4000);

     } /* end ScrSave */


static void ScrRestore (void)
     {
/*
     This function restores the contents of a screen that was saved by ScrSave
*/
     int VideoSeg;

     if (*(char far *)0x00400049 == 7)       /* Test video mode.            */
          VideoSeg = 0xb000;                 /* Monochrome video memory.    */
     else
          VideoSeg = 0xb800;                 /* Color video memory.         */
     movedata (FP_SEG ((char far *)SaveScr),FP_OFF ((char far *)SaveScr),
               VideoSeg,0,4000);

     } /* end ScrRestore */


void ScrPutS (char *String,int Attr,int Row,int Col)
/*
     This function displays null terminated 'String' with video attribute
     'Attr', beginning at the position given by 'Row' and 'Col'.
*/
     {
     register int A;                    /* Fast register storage for Attr.  */
     char far *Video;                   /* Far pointer to video memory.     */
     int VideoSeg,VideoOff;             /* Store video memory address.      */

     A = Attr;
     if (*(char far *)0x00400049 == 7)           /* Test video mode.        */
          VideoSeg = 0xb000;                     /* Monochrome segment.     */
     else
          VideoSeg = 0xb800;                     /* Color segment.          */

     VideoOff = Row * 160 + Col * 2;    /* Beginning video memory offset.   */
     Video = MK_FP (VideoSeg,VideoOff); /* Initialize video memory pointer. */

     while (*String)          /* Write characters from string until null.   */
          {
          *Video++ = *String++;
          *Video++ = A;
          }
     } /* end ScrPutS */
```

◆

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MICROCACHE™ and SPEEDPRINT™

The hottest PC utilities you can get.
Or your money back!

MICROCACHE™ monitors disk use and "learns" to anticipate your data accesses. Then, by satisfying these requests directly from RAM instead of disk, your programs run as if they are SUPERCHARGED.

- 100% Transparent Operation
- Uses any spare RAM available
- EMS/EEMS memory support
- SPEEDPRINT™ buffer option
- Requires NO program changes
- For IBM and compatibles

~~*Price $89.00*~~ ***NOW $69.00***

MasterCard VISA

Az-Tech Software, Inc.
305 East Franklin
Richmond, MO 64085

(800) 227-0644 . . . (816) 776-2700

Number 51 on the Reader Service Card

DISK FORMAT CONVERSION

XenoCopy-PC

PC-DOS program lets your PC

$79.95 + $5.00 S/H
Sales Tax if CA.
VISA MasterCard UPS COD

READ / WRITE /
FORMAT / DUPLICATE
Disks from over 300 other micros

Upgrades available from all previous versions for only $25.00 Call for Authorization

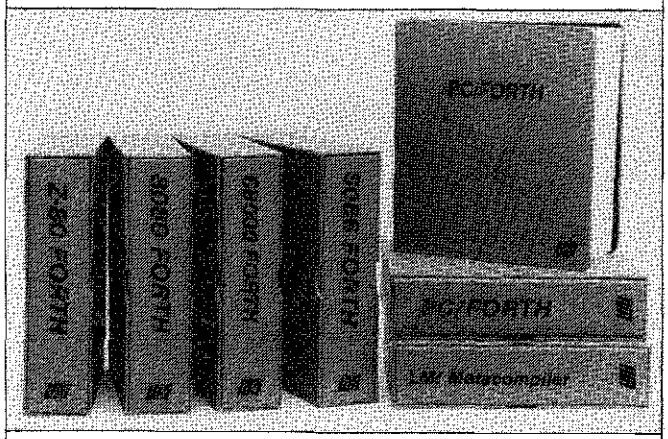
To Order Contact:

XENOSOFT™

2210 Sixth Street, Berkeley, CA 94710
(415) 644-9366

Number 123 on the Reader Service Card

TOTAL CONTROL with *LMI FORTH*™



For Programming Professionals:

an expanding family of compatible, high-performance, Forth-83 Standard compilers for microcomputers

For Development:
Interactive Forth-83 Interpreter/Compilers

- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 400 page manual written in plain English
- Options include software floating point, arithmetic coprocessor support, symbolic debugger, native code compilers, and graphics support

For Applications: Forth-83 Metacompiler

- *Unique table-driven multi-pass Forth compiler*
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, and 6303
- *No license fee or royalty for compiled applications*

For Speed: CForth Application Compiler

- Translates "high-level" Forth into in-line, optimized machine code
- Can generate ROMable code

Support Services for registered users:

- Technical Assistance Hotline
- Periodic newsletters and low-cost updates
- Bulletin Board System

Call or write for detailed product information and prices. Consulting and Educational Services available by special arrangement.

LMI ***Laboratory Microsystems Incorporated***
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to: ***(213) 306-7412***

Overseas Distributors.
Germany: Forth-Systeme Angelika Flesch, Titisee-Neustadt, 7651-1665
UK: System Science Ltd., London, 01-248 0962
France: Micro-Sigma S.A.R.L., Paris, (1) 42.65.95.16
Japan: Southern Pacific Ltd., Yokohama, 045-314-9514
Australia: Wave-onic Associates, Wilson, W.A., (09) 451-2946

Number 63 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

80286 and 80386 Programming

by
Bud E. Smith

Until now, the usual rule in writing programs to run under DOS on the 80286 and 80386 has been simple: Use the same code originally written for the 8088. This was efficient in terms of programmer time and ease-of-use for the target audience since the same program would run on any PC-compatible, no matter what processor was inside. Users of AT-class and even 80386-based machines were so happy to get increased speed that they didn't demand improved programs as well. And DOS-enforced limitations, such as the 640K limit on available RAM, made the powers of the new chips pretty much inaccessible anyway.

Now new or newly popular operating systems (OS/2 and Unix), extenders for DOS, and new programming tools allow 80286-specific and 80386-specific code to be written for programs.

In this article we'll look at some of the reasons programs automatically run faster on the more powerful processors and then at the benefits you'll get if you bite the bullet and write (or compile to) 80286- or 80386-specific code. We'll also touch on some of the effects new compilers, assemblers, operating systems, and control programs will have on the PC programmer. Understanding these considerations will help you adapt to the more complicated PC environment now emerging. The discussion will focus on real mode programming in assembly language (see the accompanying sidebar and M. Steven Baker's article on 386 protected mode programming).

Instruction	Clocks			Wall Time (microseconds)		
	8088	80286	80386	8088 (4.77 MHz)*	80286 (8 MHz)	80386 (20 MHz)
INC CL	3	2	2	.63	.25	.10
MOV BX, one word	10	5	4	2.10	.62	.25
IMUL AX, <imed. byte>	(80-98)**	21	9-14***	(16.77-20.54)	2.62	.45-.70

*1 MHz = 1 million (clock) cycles per second
**This form is not allowed on the 8088. IMUL of a word by a byte takes 80-90 clocks.
***The "Early out" algorithm finishes faster for fewer significant bits in the rightmost operand.

Figure 1. Sample instructions with clock counts and wall times for the 8088, 80286, 80386

Higher Clock Speeds

Each new Intel processor does more than its predecessor—while running at higher clock speeds. The 8088 in the PC runs at 4.77 MHz; the early 80286s run at 6 or 8 MHz (the newer ones run up to twice as fast); and the 80386, only a year old, already runs at 16 or even 20 MHz. So a program written for the 8088 will run two to four times faster on one of the newer chips due to clock speed increases alone.

This processor speed increase is lost somewhat to waits for disk access or operator input. But as multitasking becomes more popular, the waiting time will be used for work by other programs, so the speed advantage comes to the fore again.

Fewer Clocks per Instruction

It's well known that each new Intel microprocessor runs all of the same instructions as its predecessor; many of the instructions also execute in fewer clock counts than before. A new chip's higher clock speed already means that there are perhaps twice as many clock ticks ("clocks" for short) available in a given second of wall time; add to this that many instructions need fewer clocks to execute, and throughput increases substantially. Figure 1 shows the number of clock cycles and the amount of wall time needed to execute a series of instructions on the 8088, 80286, and 80386, respectively.

The assembly language programmer can do a lot to speed up programs by looking at the clock counts of various instructions. For instance, on an 8088 it's worth while to write special routines that use ADDs, or that load addresses with shifts and bit shifts for multiplying by a small integer, and that branch to an IMUL instruction only as a last resort. On the 80386, it's faster and easier simply to use an IMUL for any pair of reasonably-sized integers.

Pipelining

Both the 80286 and the 80386 use pipelined architectures, in which several things are usually going on at once:

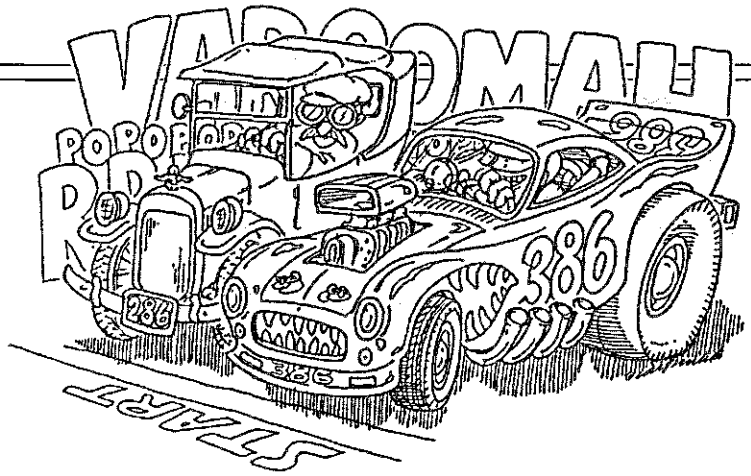
- One part of the chip uses spare bus cycles to grab instuctions from RAM and store the instructions in an on-chip "prefetch queue."
- At the same time, another part of the chip decodes instructions from the prefetch queue and stores the result in the "decode queue."
- The execution unit actually carries out the prefetched, predecoded instructions from the decode queue.

Figure 2 demonstrates how the fetch, decode, and execute units work in parallel.

One reason the 80386 is faster than the 80286 is that each prefetch (done when the currently executing instruction is not using the bus) can grab 32 bits of code, not just 16 bits. So fewer prefetches are needed to keep the queue full for the decoder, which can then keep the execution unit going at full speed.

A programmer can help or hinder pipelining in a couple different ways. The prefetcher grabs instructions in sequence; when a jump is taken or a subroutine CALL is executed, the sequence is disrupted and the prefetch and decode

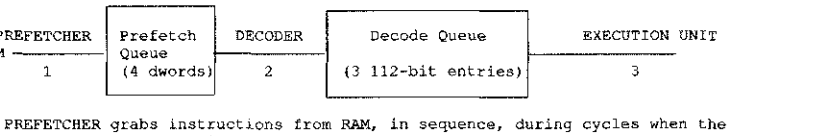
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



queues are cleared as a defensive measure to prevent the execution of no-longer-valid code. So minimizing jumps and calls helps pipelining. If the first instruction after the transfer of control is aligned on a 32-bit (dword) boundary, it can be fetched in fewer 32-bit memory accesses than if it is misaligned; your HLL or assembler may or may not contain a command to dword-align the next piece of code. After a transfer of control, the instruction unit of the 80286 or 80386 must wait while a new instruction is decoded; a short first instruction such as the ever-popular CLC runs so quickly that the execution unit must again wait for prefetching, or at least decoding, before another instruction is ready for it. So putting an easily decoded, slow to execute instruction, dword-aligned, at the top of a loop or subroutine will optimize the efficiency of pipelining.

Instruction timings for 80286 and 80386 instructions are based on the assumption that the instruction is already fetched and decoded when the processor is ready to execute it. The occasional extra cycles that occur while the fetch and decode queues fill again after a jump add a certain amount, estimated by Intel at about 5%, to the execution time of most programs.

It's easy to get carried away with optimizing code for speed when spending more time on program design or even the user interface might ultimately be more profitable. But it's worth noting that in a multitasking, or especially a multiuser, system, each program gets only part of the processor's time and must make good use of it or risk being perceived as slow.



1. PREFETCHER grabs instructions from RAM, in sequence, during cycles when the EXECUTION UNIT is not using the bus.
2. DECODER translates instruction mnemonics, etc., into a format usable by the EXECUTION UNIT. A decoded instruction can take up as many as 112 bits.
3. EXECUTION UNIT carries out the decoded instruction. Steps 1, 2, and 3 are carried out simultaneously.

Figure 2. The Fetch-Decode-Execute Cycle on the 80386

New Instructions

A single new instruction can replace several instructions or even a whole subroutine of 8088 code. We will discuss only instructions available in real (8088-type) mode—protected mode programming is a separate subject.

Several instructions available on both the 80286 and 80386 make it easier to write well behaved subroutines:

- PUSH IMMEDIATE pushes the immediate value onto the stack, replacing the normal MOV AX, IMMED, PUSH AX sequence.
- BOUND checks whether a pointer is keeping within the bounds of an array.
- ENTER and LEAVE create "stack frames" to speed entering into and exiting from deeply nested subroutines. These instructions are said to be "mostly for compiler writers," a challenge to any assembler programmer.
- POPA and PUSHA quickly pop or push all the general registers at once.

These new instructions speed things up in three ways:

1. Each new instruction can be used to replace a much longer and slower-to-execute group of instructions.

2. Less memory traffic is needed for fetching groups of instructions.

3. Coding and debugging are easier using the new instructions as simple, standard tools for arrays and subroutines.

But the really fun new instructions are found only on the 80386—the new bit-oriented test and set instructions. These will allow you to do things with bit arrays that are still technically illegal in 17 states and Canada.

The new real mode instructions on the 80386 are the bit test instructions, the set byte instructions, and a few commands for handling 32-bit data items including a long divide:

- BSF and BSR perform a bit scan of a word or doubleword, either forward or backward. The position of the first nonzero bit in the first operand is returned in the second. (Note to hardware hackers: This is

Why Assembler for the 80386?

In this article we discuss the differences between microprocessors in terms of benefits to the assembly language programmer, who can take direct advantage of the capabilities of a given chip.

A well-written compiler for a high-level-language (HLL) will create machine language optimized for a specific processor. But the degree to which a given compiler takes advantage of a given new instruction varies.

It's possible to use an HLL for initial coding, disassemble the resulting object code, then optimize the result by hand. But many of the new optimizing compilers produce object code that is possibly better than a human could quickly produce, but is certainly harder to follow, making optimization by hand difficult.

The best combination of ease of writing, ease of maintenance, and access to the full capabilities of a given processor may be to write the program in an HLL, then rewrite speed-critical and processor-dependent pieces of code in assembler. With a truly top-notch compiler, the HLL can be used for some or all of the work that would otherwise be done in assembler. But if you are already good at "real" programming and you have a large library of assembler routines at hand, the mixed HLL-plus-assembler approach may be the best solution for a long time to come.—*Bud Smith* ♦

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Better 80386 Programs

Those poors souls who must program in constricted environments (CP/M, the Apple II, PROMs, etc.) often direct a "Repent, for the end is near" attitude towards those DOS programmers who write large, sloppy programs that use RAM as if there were no tomorrow.

Well, tomorrow is here, in the form of OS/2 and control programs for the 80386. There are two main ways in which these new factors make small, tightly written programs look good:

1. Virtual Machines

Under the DOS "control programs" now coming on the market, multiple applications can be run at once, with each running as a separate task or in its own "virtual machine." In a certain amount of memory, a user may be able to run three large, sprawling applications or four to five equally useful, tightly written ones. Guess which programs will get more use—and a reputation for efficiency?

2. Virtual Memory

Virtual memory is used by OS/2 and some 80386 control programs. In virtual memory, some of the code and data that would normally be in RAM is stored out on disk until needed. Under OS/2, 64 Kb-sized segments are swapped in and out; under an 80386 control program, 4 Kb pages are moved back and forth.

A pause occurs each time a program's execution causes it to jump to code or access data in a "swapped-out" (disk-stored) page or segment. A small, tightly written program will run with few such faults while a more sprawling program may cause the operating system to spend much of the time moving code and data from RAM to disk. (Note that assembler programmers have more control over code and data structures than HLL programmers.)

OS/2 and 80386 control programs are now largely in the hands of a small but influential group of programmers and power users. This group can be expected to voice its opinions loudly and frequently, pasting long-lasting labels, for good or ill, on certain programs and the companies that produce them.—*Bud Smith* ♦

the software version of the hardware function, "priority encode:.")

- BT, BTC, BTR, and BTS are all "bit test" instructions, which use their second operand as a pointer to a bit position in the first operand. The Carry flag is set if the indicated bit is set. Depending on which instruction is used, the tested bit is left unchanged, complemented (inverted), reset (made 0), or set (made 1), respectively. Intricate, easily updated bit tables can be constructed with these new instructions.
- In SETcc, cc indicates a "condition code" determined by the current settings of one or more flags. If the flag(s) has the status indicated by the command, the byte pointed to by the single operand is set to one; else it is cleared. This instruction type is a bit obscure; to spark interest, perhaps Intel should have a contest with a prize for the most creative use of SETcc instructions. One possible application is to transform code with lots of Jcc's into code with several SETcc tests and a CASE-type structure at the end.
- CWDE, CDQ, SHLD, and SHRD will convert a word to a doubleword, extending the high bit; convert a doubleword to a quadword; or do a double shift either left or right (a shift where vacated bits in one operand are filled with values from a specified other operand). This is basically bookkeeping-type stuff for dealing with 32-bit data items.

Unless you write Ada compilers for fun, the most interesting instructions are the new bit instructions. It's too bad graphics coprocessors are becoming so popular because these new instructions and the absence of forced segmentation on the 80386 make it capable of handling large video displays. No matter—these instructions make it possible to write arrays that boggle the mind. The largest possible 80386 segment is 4 Gb; there are eight bits per byte, so an array with over 32 billion positions is possible. Short of that, programmers will no doubt come up with incomprehensibly complex multi-level indexing and other arrays before long. The new instructions make it possible to do operations nearly or completely unmanageable before in a few lines.

Speed Increases for the 80386 Only

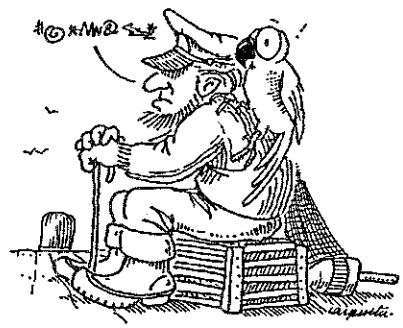
The 80386 is a true 32-bit microprocessor—it has 32-bit addressing and 32-bit data paths, internally and externally. Using 32-bit data allows fast access to large data items or a group of smaller ones. This will decrease the time spent accessing data and increase the number of records, fields, etc., that a given program can support.

In protected mode (breaking my promise not to touch on that), segment size is programmer-definable, with a maximum size of 4 Gb per segment. A common programmer response to this opportunity has been to set all segment bases to 0, set the top of each segment to 4 Gb, and use the 4 Gb space as flat, unsegmented memory—as used on the 68000 series and other processors. The harsh limitations enforced up to now by the 64 Kb segment limit simply disappear. For programmers of number-crunching applications, this single change is reason enough to switch over completely to the 80386.

Several compilers that support huge arrays by using large segments—especially FORTRAN compilers—are now coming out (see Miscellaneous, *PJ*, Volume 6.1, January/February 1988, and M. Steven Baker's article in this issue). The compilers incorporate DOS extenders that allow DOS programs to freely mix 16-bit, real-mode routines with 32-bit, protected-mode routines. If you currently have large arrays in your programs, they will run much faster once the end-of-segment checking is removed; if you've been forcing all your arrays to stay under the old limit, you can now run wild. ♦

Bud E. Smith is the author of Programming the Intel 80386 *and other books and articles on PC hardware and software. He can be reached care of*

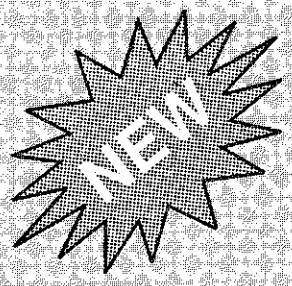
IGC
4800 Great America Parkway
Santa Clara, CA 95054



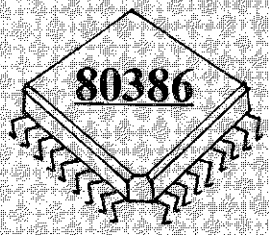
·CURSOR·

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

SERIOUS DEBUGGING AT A REASONABLE PRICE



Soft-ICE™



All the speed and power of a hardware-assisted debugger at a software price - $386

Features

Real-time break points on:
- Memory locations
- Memory ranges
- Execution
- I/O ports
- Interrupts (hardware and software)

Dual monitor support

System memory map

Regain control with a keystroke
Even with the following code:

```
CLI
MOV   AL,0FFH
OUT   21H,AL
JMP   $
```

and much, much more

How Soft-ICE works

Soft-ICE unleashes the power of the 80386 to surround your program in a virtual machine. This gives you complete control of the DOS environment. Soft-ICE uses 80386 protected mode features, such as paging, I/O privilege level, and break point registers, to add real-time hardware-level breakpoints to your existing DOS debugger. To use Soft-ICE you simply pop the Soft-ICE window up through a key sequence, set your hard break points, then return to your soft debugger. As the target program is executing, Soft-ICE recognizes when the breakpoint conditions have been reached and gives control back to your soft debugger. And this is all done at full 80386 speed! Soft-ICE can also be used in stand-alone mode. This comes in handy if you are debugging loadable device drivers, interrupt handlers, or terminate and stay resident programs. All of the standard debugging features are available to help you find the most difficult systems problems.

Benefits

- **Works with CodeView** --To get you up and going as fast as possible, Soft-ICE is designed to work with your existing software debuggers such as CodeView and Pericope I & II.
- **Breaks the 640K barrier** -- If you have extended memory, Soft-ICE takes up **ZERO** bytes of memory in the first 1MB of address space. This means you can load and debug your largest programs.
- **Power of an in-circuit emulator** -- At only $386, you can give every member of your software development team the power of an in-circuit emulator.
- **AT compatible 80386 PC's** -- Soft-ICE works with all AT compatible 80386 PC's, such as COMPAQ's Deskpro 386 and the IBM Model 80.
- **Easy to learn** -- Soft-ICE is so easy to learn that you can be finding bugs with Soft-ICE by the time you could install a hardware-assisted debugger.

"Since Soft-ICE doesn't take up any of my memory I have it in my AUTOEXEC.BAT to load every day... It has saved me at least one months time on my latest device driver program." Peter Ricker, President of Maverick Software

30 day money-back satisfaction guarantee. Visa and Master Card accepted. Ask about our coupon program.

To order or to get more information, call (603) 888-2386

NU-MEGA TECHNOLOGIES

P.O. BOX 7607
NASHUA, NH 03060-7607

Number 160 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

80386 Software Development

by
M. Steven Baker

Last issue in *PJ* Miscellaneous, I introduced several 80386 DOS compilers and development tools. Several minicomputer vendors have been porting their optimizing 32-bit Unix compilers to the 80386. I'll focus on their DOS versions rather than their products for Unix/Xenix on the 386. This article looks at several 386 compilers and runtime environments and provides limited benchmarks to illustrate the potential speed advantages of certain programs on this new hardware. Results for numerically intensive programs using the Weitek math coprocessor chip will have to wait for a later issue of *PJ*.

The compiler and runtime products reviewed appear in Tables 1 and 2. They include three 80386 DOS extender programs and debugger runtime environments and several C and FORTRAN compilers. In addition, I used an 80386 assembler, linker, and librarian from Phar Lap and an 80386/387 Hummingboard (coprocessor hardware board) for the PC from A.I. Architects.

Table 1. 80386 Runtime Environments Reviewed

	PharLap	IGC	A.I. Architects
Product	386|DOS Extender	X-AM	OS/386
Version	1.1V	1.1	1.9.09
Price	$995	$595	$495+

The MetaWare C and Pascal compilers (I tested only the C compilers) are reasonably mature products, having been available for about a year. The three 386 DOS runtime environments are more recent and are changing rapidly to meet market demand. For the 386 FORTRAN compilers, this is really a preview. I used mostly Beta test copies of these compilers, and my comments should be viewed in this light. The AI Hummingboard 80386/387 coprocessor has been out for about a year and appears to be a stable product.

The software tests were conducted in two environments: (1) a 16 MHz Compaq 386/387 with a 70 megabyte hard disk and 2 megabytes of memory and (2) an AT&T PC6300 (with Nec V30 chip) with a Seagate 277R (60 megabyte) hard disk using the A.I. Architects 20 MHz 80386/387 Coprocessor board with 4 megabytes of memory.

Why Bother with the 386 Anyway?

That's what the CP/M Z80 and 8085 folks said about the 8086. Time marches on, and in hardware and software, product differentiation and marketing are driving forces. As long as speed and power spur marketing, the powerful features of the 80386 just can't be neglected. The 386 DOS extenders serve in the interim to allow large 386 applications to be run under DOS until a true 386 operating system (Unix, OS/3, CDOS-386, VMS) is well established. In November, I felt that these DOS extenders would have maybe a two year lifetime. Others, including the developers of these products, agreed, but as the muddling wars continue, with OS/2's various versions and extended editions, I now feel that the life of these products may be much longer, perhaps up to five years.

80386 DOS Runtime Environments

The 80386 can almost be considered several different chips in a single 132-pin package. Most 386 PC machines are running the 80386 chip in its real mode to emulate the 8086 chip. Actually, the 80386 handles the LOCK prefix and the PUSH SP instruction differently, but these aren't normally used by DOS applications. In real mode, memory is limited to 1 megabyte and some instructions are not available. On the plus side, DOS programs run just fine and much faster. On the negative side, programmers have to deal with those ?#@ 64K segments of the original 8086. But the 386 has other modes: (1) a protected mode (native mode) that supports both 286 and 386 memory protection and segmentation and (2) a virtual 8086 mode to provide emulation of the 8086 when in protected mode.

Only the protected mode takes full advantage of the power of the 386. In protected mode, the physical and virtual memory space is seemingly unlimited (4 gigabytes and 16 terabytes). (Oops—back to the real world, where available hardware limits physical memory to 16 megabytes or less). In the protected mode on the plus side is this large memory space. On the negative side (depending on your perspective), memory protection is enabled—you can no longer write into code segments, muck about with interrupt vectors, and do other fun things. And to use the protected mode, software must first set up the prerequisite memory management tables, including segmentation and paging tables if paging is enabled (the 386 has built in support for 4K physical to logical memory paging). These include (in Intel jargon) global (GDT), local (LDT), and interrupt descriptor tables (IDT). Setting one bit in the LDT for a program converts all 16-bit register accesses to similar 32-bit instructions. For example, the instruction opcode that meant PUSH AX (push the 16-bit register onto the stack) now means PUSH EAX (push the 32-bit register). The most common memory model in this mode is a flat one (the small model) with all segment selectors pointing to the same chunk of memory. No more segments—just 4 gigabytes of 32-bit address space! All is not free, however; memory management imposes some software penalties on 386 instruction times. Because of memory protection, instructions that modify segment registers take longer in protected mode. Enabling paging can also slow down memory access. Although the 80386 has a built-in, on-chip cache of 32 most recently used page table entries to speed memory access, that represents only 128K of memory.

Table 2. 80386 DOS Compilers Reviewed

	MetaWare	MetaWare	LPI	MicroWay	SAIC
Product	High C	High C-386	LPI-FORTRAN	FORTRAN-386	SVS FORTRAN
Version	1.4	1.4	02.07.03	1.3B	2.7
Price	$595	$895	$695	$595	$895

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Intel's simulations suggest that this should catch about 98% of normal memory accesses, but this depends on the code. If a page table entry is not in the cache, the 386 must read two double word entries from the page translation table in memory before accessing the memory location of interest.

In addition, some gateway or bridge to DOS must be constructed—the task of the 386 DOS runtime extenders. These control programs set up the required memory management structures including a gateway to DOS and other operating system and hardware services, load a 386 program into memory, and start it. In general, these runtime extenders set up an emulation of the DOS and BIOS calling conventions using INT instructions. They intercept DOS calls, transform them for passing to the real DOS kernel that waits in low memory, pass the call to the real DOS and BIOS, and pass the returned information back to your 386 program. The subtle differences are in this emulation of DOS and BIOS that your application sees and in facilities to allow a 386 protected mode program to call an 8086 real mode procedure (say a graphics routine or a TSR) sitting down in low memory, load other 80386 programs, and provide helpful development environments.

Some General Observations

I worked with the three major 386 DOS extenders. They all include a debugger, utility programs, and examples of their use to create and execute protected mode 386 programs. My first observation relates more to the programming process than to the specifics of these various runtime environments. Let's take a typical application and divide the time spent on various tasks. You might spend 20% on planning and design, 40% for programming, and 40% for debugging. When you decide on a runtime environment, you are also choosing a debugger and technical support. Because of software differences among them, a debugger from one vendor will not work under another 386 environment.

All three DOS extenders emulate DOS and the BIOS to a substantial degree. If we rank runtimes, A.I. Architects' OS/386 is the closest to DOS while IGC's X-AM is the furthest. The Phar Lap DOS Extender lies somewhere in between. The three runtimes support three different executable file formats, and Phar Lap has proposed two more. From a programmer's perspective, I wish these three runtimes were closer together in their DOS and BIOS emulation and EXE file formats and in the way they handle other hardware issues (e.g., the Weitek chip). I'd rather choose 80386 compilers and development tools without being forced to make a choice of development and runtime environments at the same time. With the exception of the MetaWare compilers, if I choose a 386 compiler, I'm also unfortunately choosing an associated DOS extender and debugger.

These DOS extenders are not ideal candidates for fast file I/O. Whenever a DOS call is made by a protected mode application, the machine state is saved twice—once by the DOS extender and once by the underlying DOS system. Also, under normal circumstances, file I/O is handled by the real DOS and BIOS down low in physical memory and block moved by the DOS extender to your program's disk buffers in high memory. This additional block move and state save is bound to impose a performance penalty on file I/O. Therefore, you're best off minimizing DOS calls and doing I/O in large chunks if possible. Onward to specifics.

A.I. Architects OS/386

A.I. Architects (AIA) sells a Developer's Kit that includes the OS/286 or OS/386 kernel, a symbolic debugger, and an EXPRESS utility to generate a protected mode executable file from Microsoft or Phoenix DOS linker output. The runtime included in the Developer's Kit can be used solely for development. Runtime licensing starts at $40/each and decreases with quantity for distribution of the OS/386 runtime with your application. AIA's OS/x86 runtime kernel differs substantially from the other vendors'. OS/286 supports a multiple segmented memory model on the 286, allowing applications to break the 640K memory barrier under DOS. Language support and library files are provided for use with Microsoft C and FORTRAN 4.0, MetaWare High C, Lattice C, and Lahey F77L-EM to create 16-bit protected mode programs. The same memory map is also supported on the OS/386 runtime along with the more common flat, unsegmented (small) memory model. AIA's protected mode programs use the file extension EXP (EXecutable Protected mode) for both 16-bit and 32-bit programs. Unfortunately, this is the name adopted by Phar Lap for their default executable files even though the files have different structures. AIA calls the Phar Lap 32-bit files PLX (Phar Lap eXtended) and can run them. A strength of the AIA product is its support for larger applications under DOS on the 286.

Among the three runtime environments, OS/386 offers the closest emulation of DOS and BIOS. Its manual is the clearest of the three DOS extenders', describing both compatible and slightly incompatible DOS calls supported. Primary variances from DOS are that FCB I/O calls support records up to 16K, DOS memory allocation, and EXEC calls. Otherwise, all DOS calls are fully supported except that 32-bit registers are used.

OS/386 switches from protected mode to real mode for DOS calls and does not use paging. Only memory above the 1 megabyte barrier is available for use in the protected mode. AIA based their decision to forego paging for the time being on a bug in current versions of the 80386 with 80387 math chip errors (the infamous Intel ERRATA#21). The Weitek math chip is to be supported in the next version. Extended DOS calls are supported (**AH >= E0h**) for calling real mode procedures, setting arbitary interrupt vectors, creating code and data segments, getting segment information, and block transfers to low memory. Execution of a 386 program is preceded by the UP command on the command line. The AIA debugger (CP—Command Processor) is a powerful command shell that can execute DOS-like built-in commands and batch files, as well as be used as an assembly language symbolic debugger. The shell includes a history command and a built-in command-line editor using key bindings similar to the standard EMACS editor defaults. The shell has a powerful macro processor that allows invoking a batch file of macros to make the debugger or shell look similar to SYMDEB—although other user interfaces are only a macro file away. This shell is easily the most powerful of the debugging environments provided with the runtimes. Currently, the only 80386 compilers that work under the OS/386 runtime and produce 32-bit code are MetaWare's C and Pascal. The next revision of OS/386 should support LPI and Green Hills compilers.

IGC X-AM Development Environment

Last May, IGC acquired the 80386 development team from Softguard Systems (the copy protection folks). IGC now offers two 80386 system products—VM/386 (a powerful multitasking control program similar to Microsoft's Windows-386 and Quarterdeck's DESQview) and X-AM, their 386 DOS extender. Of the multitasking programs I've seen so far, VM/386 looks very powerful. X-AM is its lesser cousin. The IGC X-AM (eXtended Address Mode) runtime, called VM/RUN, is a different model of the machine than OS/386. The VM in these product names stands for Virtual Machine (the IGC development team comes from the IBM mainframe VM world). Although neither of these products currently implements virtual memory, the overall design concept is based on creating virtual machines with complete protection of the operating system from damage by an errant program.

The other two DOS extenders run DOS in real mode and 386 applications in protected mode. VM/RUN puts the entire system in protected mode and runs the DOS kernel in lower memory in the Virtual 8086 mode. VM/RUN provides a completely flat memory model for the system, with applications loaded at the 32 megabyte address. No segment registers are used. DOS is down low in memory where you would expect it. For example, monochrome screen video memory is mapped at B0000h, and BIOS data at 400h—their actual physical addresses. All of this is done with paging. VM/RUN is the largest DOS extender (200K) and takes up the most memory. After it's loaded and has allocated buffers for I/O, EXECing other ap-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

plications, and TSRs, memory below 640K is pretty much used up. These memory protection features are justified for VM/386; their necessity for the single-tasking X-AM system is arguable. X-AM supports the Weitek math chip memory-mapped up high in the 386 address space at the same location Unix chose on the 386. X-AM looks the least similar to DOS—FCB I/O calls are omitted, and a number of other functions are different. On startup, X-AM passes a global data structure (GDA) with system information to your application.

X-AM requires a REX (Relocatable EXecutable) file extension for 386 programs. The REX file can be generated by the Phar Lap 386 linker and includes relocation information. The runtime consists of four REX files and a main (VMRUN.COM) loader file. The REX files must be available along the current path. For debugging, a special debug REX file must be substituted for one of these standard REX files. The main COM file is a program loader used to make profiles for the executable 386 program. Similar to a feature found in Unix, the loader uses **arg[0]** (the name of the command invoked) under DOS 3.x to determine the REX file to load. A more recent version of the runtime concatenates these files, but this then requires an additional 110K loader file with the same name as your 386 REX application file. This makes a lot more sense under Unix, where multiple links (with different names) to the same physical file can exist. Under DOS during the development stage, each one of your REX programs carries around this extra file.

One awkward feature of development with X-AM is that the stack is completely protected (the IGC model of the machine). The COM file has locations that may be changed that determine the amount of stack to allocate, whether the debugger should be available, etc. The stack does not grow from this default value. After creating a REX file, the Make Profile utility (MP) must be used regularly to modify these default values. IGC is working on correcting this problem in the next revision. The assembly language debugger is the weakest of the three products'. It is "window oriented" if you consider a fixed line across the top of the screen with a register display and a fixed line across the bottom of the screen with a prompt for your commands a windowing system. The 16 lines of the display area are used for dumps and unassembly, but this is done in a maximum of only 16 line dumps at a time. The X-AM manual I received was also the weakest of the three runtimes'—no index and an incomplete list of error messages. The Weitek chip was not mentioned (at least I didn't find it) although I know it's supported. The X-AM environment is currently supported by compilers from MetaWare, SVS, and LPI.

Phar Lap 386|DOS Extender

Phar Lap includes in its 386 assembler/linker package RUN386, their 386 DOS Extender. This version can be used solely for development. In order to distribute this runtime to end users of your product, you must buy a runtime license ($995). The unlimited runtime license comes with BIND386 and CFIG386 utilities. The binder binds the 386 system kernel to the front end of an application, turning it into a DOS EXE file—really a mere concatenation process. Under development, a 386 application is invoked by prefixing the command line with RUN386 followed by the application to invoke and its command line. RUN386 supports two different file formats—the Phar Lap EXP and the IGC REX. Although RUN386 can load a REX file (Phar Lap wrote the assembler/linker currently used by all three vendors), it cannot in general execute REX programs because of differences in the Phar Lap and IGC runtime environments. I was not able to execute REX files supplied with or created by the SVS FORTRAN compiler with RUN386.

The Phar Lap runtime is a flat memory model with several hardwired segment selectors for memory mapping. For example, your program CS has a segment selector of 0Ch while DS and ES have a selector of 14h—all pointing to the same block of physical memory. RUN386 uses about 100K of low memory aside from DOS, leaving about 500K of low memory plus any extended memory above 1 megabyte available for protected mode use. RUN386 uses memory paging on the 386. This is the only DOS extender that easily makes any sizable quantity of low memory available for 386 programs. This may be a consideration for low memory (2 megabyte) machines.

The DOS protected mode function calls are reasonably similar to normal DOS calls with a few exceptions. Of course, registers used are 32-bit instead of 16. The DOS **get_version (AH=30h)** is used to return information about the runtime. The DOS **getint (AH=35h)** and **setint (AH=25h)** calls are not supported. FCB type file I/O calls are not supported. Memory management and EXEC calls are slightly different from their DOS counterparts. The Weitek math chip is supported by mapping it into a 64K block of memory pointed to by segment register GS (the 386 has six segment registers—FS and GS and the 4 registers available on earlier chips). This requires a segment override and address length override prefix instruction in order to access the math chip although timing and opcode length end up the same when accessing the math chip with a full 32-bit address. But loading a register with the start of the Weitek chip address and using register indirect addressing would require an extra segment override byte.

The latest versions of RUN386 are to incorporate support for DESQview and the Compaq Extended Memory Manager (CEMM). Applications developed with Phar Lap's DOS Extender will be able to multitask under DESQview for the 386. The Beta test version (1.1W) had a few bugs, so I subsequently did my testing with Version 1.1V, which does not include DESQview support.

RUN386 comes with a solid 200-page manual describing in detail all DOS and BIOS functions supported. To registered users, Phar Lap sends out free software updates and a technical newsletter documenting changes in all of their products. The Phar Lap debugger 386DEBUG is not included with the assembler or runtime and must be purchased as a separate product ($195). The debugger feels much like Microsoft's SYMDEB, featuring many of the same commands and support of symbolic names. The MetaWare 80386 compilers, along with LPI and NDP compilers, run under and generate code to operate using Phar Lap's DOS Extender.

MetaWare High C and High C-386

MetaWare was the first to supply compilers under DOS for the 80386, with Pascal and C compilers out for over a year now. These products are currently the most mature in the DOS environment (they support DOS's quirks). MetaWare offers two sets of compilers—High C and Pascal for creating DOS programs and High C-386 and Pascal-386 for protected mode 80386 programs. The 386 compilers could almost be considered cross compilers since they operate on an 8086 (or the 80386 in the real mode) and do not require an 80386 for compilation. This is a convenient feature because they can be run under the normal DOS environment and aren't limited to a particular 386 DOS extender. Without these compilers, most of these runtimes wouldn't be available. Both the AIA and Phar Lap 386 DOS extenders and debuggers were written in MetaWare C—a pretty strong recommendation. IGC wrote their runtime and debugger in assembly language although you wouldn't know it by the executable size.

MetaWare supports all three 386 runtime environments fully, and the source code to the INIT module is provided for customization. MetaWare also provides the source code for other parts of the runtime for embedded applications. The compiler generates code for the 80287, 80387, and Weitek chips and comes with libraries for the math chips and an emulation library. These compilers are highly optimizing and offer excellent code generation. A large array of command-line and inline **pragma** options in the source code are supported, as well as the current draft of ANSI C including prototyping, the new preprocessor directives, standard header files and libraries, and not promoting floats to doubles. This last option provides a means for C to compete in the numerically intensive programming market generally dominated by FORTRAN.

The 8086 DOS C compiler supports code generation for the 8086, 80186, 80286, NEC V30, and 80386 CPUs. In the real mode, certain 80386 instructions are used for long int divides, block moves, bit routines, etc., but memory addressing is still limited to segment:offset addressing. This is more a limit of the existing DOS linkers and EXE relocation schemes than of the compiler technol-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ALL GAIN, NO PAIN

Blow away the 640K barrier

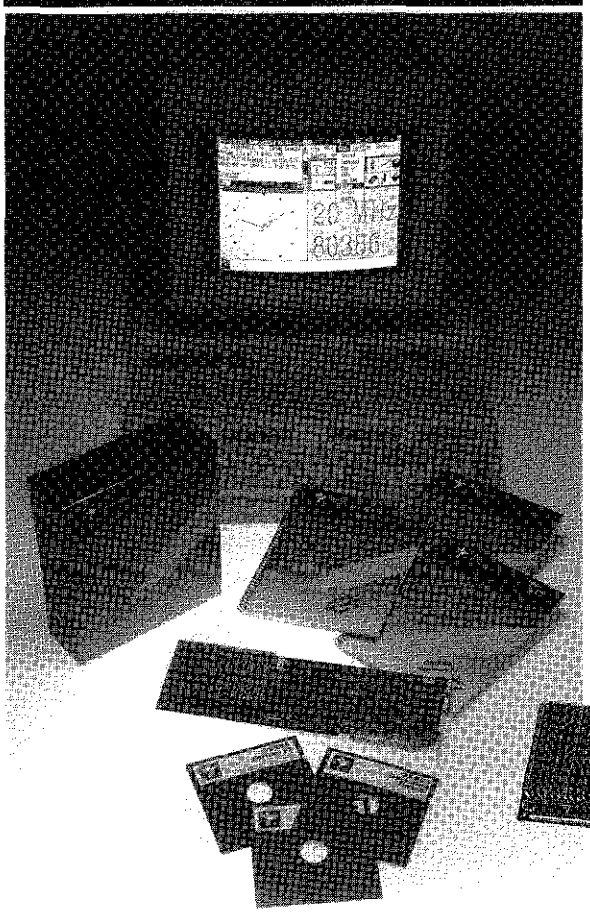
Gain the benefits of protected mode the easy way with OS/286™ and OS/386™. These tools for C, Fortran, Pascal and Assembly language programmers permit rapid conversion of existing DOS applications from "real" 8086 mode to "protected" 286 and 386 mode. They don't replace or modify DOS, but extend it to protected mode.

OS/286 and OS/386 are the only DOS extenders that span both the 286 and 386 processors, with 32-bit capability *today* on 386s that yields twice the performance of 16-bit mode. OS/286 and OS/386 have quickly become the preferred solution for developers of high performance, memory-intensive applications, including CADKEY, CASE, and Gold Hill, and premier language developers Lahey, and Metaware.

Our optional TOUCHDOWN™ BIOS supplement provides fast and reliable protected mode operation on *any* 286 system, even those with problems resetting the 286. (Ever notice how few existing machines Operating System/2 runs on?)

If your applications are running out of memory or need more speed, don't wait for the "solution" that means abandoning your investment in DOS. Enhance them now with OS/286 and OS/386 — *products not promises.*

MAKE BIG PROGRAMS RUN FASTER IN PROTECTED MODE



OS/286™ & OS/386™ Benefits:

- Gain multi-megabytes of directly addressable memory (15-Mb-286, 4Gb-386)
- Increase performance by eliminating overlays and EMS
- Convert in days, not months
- Continue to work with a DOS interface and use existing TSRs, device drivers, graphic routines, etc.
- Stay compatible with the widest array of systems

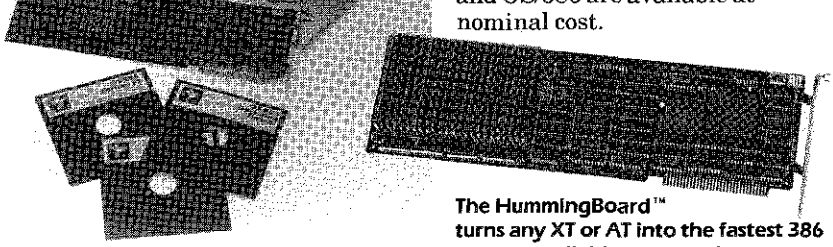
A.I. Architects Software Developers Kit **$495**

includes full support for:

- MetaWare High C (16 & 32 bit)
- Professional Pascal (16 & 32 bit)
- Lahey F77L FORTRAN
- Microsoft C & Fortran 4.0,
- MASM, MS-Link
- Phoenix PLINK86
- Halo & GSS Graphics
- Pharlap 386: ASM/LINK

more to come

Run time licenses for OS/286 and OS/386 are available at nominal cost.



The HummingBoard™ turns any XT or AT into the fastest 386 system available. The dual processor architecture boosts performance significantly over comparable single processor systems or accelerator boards. Available with 2 to 24Mb RAM, 16 or 20Mhz speed, and 387 floating point coprocessor.

A.I. Architects, Inc.

One Kendall Square, Cambridge, MA 02139

TEL (617) 577-8052 FAX (617) 577-9774

OS/286, OS/386 and HummingBoard are trademarks of A.I. Architects, Inc., High C and Professional Pascal are trademarks of Metaware, Inc., F77L FORTRAN is a trademark of Lahey Computer Systems, Inc., Microsoft and MS-DOS are trademarks of Microsoft Corp.

Number 180 on the Reader Service Card.

PRODUCED 2006 BY IMSZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Table 3. Standard C Benchmarks

	Hummingboard 20MHz 80386/387			AT&T PC6300 8MHz V30	
	MetaWare C-386	Metaware C	Microsoft C	MetaWare C	Microsoft C
Dhrystones(k/sec)	8,116	4,374	4,540	878	890
Execution Time (secs)					
Fibonacci	38.1	51.7	54.3	256	216
Sieve	4.2	6.9	6.7	41	42
8 Queens	3.6	5.8	7.4	35	46

NOTE: MetaWare C and Microsoft C 5.0 compiled for large model and 80186, with no stack probes and optimizations to simulate larger programs.

ogy. The MetaWare High C compiler (DOS) also supports both the Phoenix PLINK86 linker and the INTEL object module format for embedded applications or other environments. High C also supports linking routines from about the widest variety of other compilers I've seen—from most Microsoft languages to Lattice C. Various **pragma**s specify the order of arguments, who pops arguments, and the registers used to return values. The compiler can generate object files compatible with Microsoft's CodeView debugger and the INTEL PSCOPE-86 debugger.

The High C-386 compiler generates object code for the Phar Lap 386|LINK program using the easy-OMF format as defined by Phar Lap. This format has become a defacto 386 linker standard under DOS. Both compilers support a variety of options—to enable and disable various optimizations, generate a mixed pseudo assembly language and C source listing, enable prototypes, emit line numbers and procedure names for post mortem debugging—the list goes on. The compilers may be configured using a utility with some of these options as defaults. In addition, the compilers support a handy feature given all these options—a Profile file. Upon invocation, the compiler looks for a file called HC.PRO (a command-line option can change the Profile filename to be used). The Profile file can contain **#def**ines, **pragma**s, prototypes, etc., necessary for a particular compilation. This is convenient given the large number of options, and it's also nice when porting software from one environment to another. In this latter case, main source code modules can be maintained with target dependent compiler options placed as **pragma**s in the profile. This is much more convenient than trying to squeeze this information into the DOS environment space as is done with Microsoft C and other compilers. Your C code is often a lot more readable without the myriad **#ifdef**s and **#else** clauses. MetaWare also supports using the DOS environment space for the location of include files.

Documentation includes a *Program* manual describing compiler operation and use, a *Library Reference* manual, and a *Language Reference* manual. Each is about 200 pages with an individual table of contents and index. I have one complaint with the *Library Reference*: The material is organized into sections by the library header file. I would much rather see an alphabetized list of library functions by name. The library is fairly complete, but not as comprehensive as the Microsoft C 5.0 libraries for DOS. Certain Unix compatible DOS library routines are not currently provided for file time stamping, mode setting, and determining file size. A set of some low-level Unix I/O calls is provided in a separate file, but these are not documented in the *Library Reference*. MetaWare does provide object file and library support for FlexOS, Concurrent DOS, and some other 80x86 system environments.

I used some standard integer and floating point benchmarks to get a feel for the impact of moving from the real mode to the protected mode. Table 3 shows a comparison of execution results of simple integer benchmarks using High C and Microsoft C 5.0 and the High C-386 compiler. In the best case for these simple programs, putting the 386 into its 32-bit mode about doubles execution speed. The performance difference of about 500% is more dramatic going from the AT&T PC6300 with 8 MHz V30 to the 20 MHz Hummingboard coprocessor. Comparing High C with Microsoft C 5.0 for some of these benchmarks, each is faster. For floating point, the results indicated about a 100% speed up in going to the 386 protected mode. Using the Weitek math coprocessor with the 80386 as supported by MetaWare would show even more dramatic improvements. Be mindful that these benchmarks are all for small applications that don't require overlays. As soon as we squeezed a large application into the DOS real mode memory limitation with overlays, the comparison with a protected mode program would become dramatic indeed. I was able to port 32-bit C code from the GNU project using C-386 when the DOS compilers would not execute it properly. It's surprising how much supposedly portable C code is dependent on integer size.

LPI FORTRAN

Because quality C and Pascal compilers have been available from MetaWare, most vendors have focused on FORTRAN and other languages for their first releases on the 80386. Language Processors (LPI) has ported their FORTRAN, COBOL, and BASIC compilers to the 80386 under DOS. LPI has already ported the same compilers and PL/I, RPG C, and their source code debugger to Xenix and Unix on the 80386. LPI FORTRAN currently supports the 80287 and 80387 math chips, and Weitek math chip support is in the works. The LPI compilers are written in PL/I. The user has access to the compiler developers for support. LPI has tried to structure their compilers to look much like their Unix products and support using several of their compilers. The FORTRAN compiler executes under the Phar Lap 386 DOS Extender and produces code for either the Phar Lap or IGC 386 runtime environments. The compiler is bundled with the Phar Lap linker and runtime. The code produced senses at runtime which DOS extender it is operating under—an attractive feature. The compiler generates object files directly for the Phar Lap linker.

LPI FORTRAN supports options for optimization, listing control, cross reference, floating point, SAVEall, etc., on the command line. Three levels of optimization may be turned off in stages. This is a handy feature since probably 90% of porting problems are from compiler code optimizers. When you move a mainframe program, say, between an IBM and a VAX, the first thing that's usually done is to turn off all optimization. The LPI FORTRAN compiler can also generate a listing file with pseudo assembly language statements. The generated assembly language does not use standard Intel mnemonics. LPI supports a convenient feature whereby the command-line options and file names to the compiler may be taken from a file—allowing you to get around the DOS command-line limit and build an option file to literally compile and build a 250-module program in one swoop. LPI's post processor then builds the LNK file for the Phar Lap linker.

The documentation includes a *User's Guide* and a *Language Reference*, each with table of contents and index. The *User's Guide* is short but adequate, and does briefly cover mixed language data type considerations for linking with other LPI languages. LPI's compiler operated more slowly than the other 386 FORTRAN compilers. This was partly a result of LPI's decision to mimic the LPI Unix compiler interface, and they plan to revisit

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Table 4. Standard FORTRAN Benchmarks

	80386 DOS Compilers			DOS Compilers			
	LPI	MicroWay	SAIC	Lahey	Austec	Definicon	VAX
WHETSTONE							
Execution time in k whets/sec							
single prec.	1274	1557	1671	xxx	765	1285	1751
double prec.	1053	1348	1481	xxx	700	1135	1183
LINPACK							
Execution time in mflops/sec							
single prec.	.14	.18	.20	.06	.12	.13	.35
double prec.	.14	.15	.16			.11	.14

NOTE: DOS compilers tested on Compaq 16 MHz 80386/387; VMS FORTRAN on VAX 11/785+ with FPA, and Definicon 20 MHz 68020/68881 using SVS Compiler

this decision. The LPI compiler generated fast code, but was not quite equal to the other 80836 compilers. The optimization level had to be reduced in order to get the Linpack benchmark to return correct results.

MicroWay NDP FORTRAN-386

MicroWay has been working with minicomputer compiler vendor Green Hills Software (Glendale, California) to provide Green Hills' family of C, Pascal, and FORTRAN compilers under DOS for the 386. MicroWay has traditionally specialized in math coprocessor hardware and software for the PC. They have been working with Green Hills to add Weitek support and other DOS features. These compilers are globally optimizing—they require lots of memory for compilation. The compilers are written in Pascal. FORTRAN-386 supports the 80287, 80387, and Weitek chips and comes with libraries for each math chip. FORTRAN-386 executes under the Phar Lap 386 DOS Extender and produces code for this same runtime environment. The compiler generates assembly language files for the Phar Lap assembler, but this additional step slows the speed of compiles. The Phar Lap assembler and linker must be purchased separately.

FORTRAN-386 supports a large range of command line options and can use part of the 80387 stack or some of the Weitek registers for register variables. The compiler is also conservative about using variables in registers when other procedure calls take place. This compiler demonstrated the case in which creating temporary FORTRAN variables can generate faster code because of the register allocation scheme. The excellent compiler documentation includes a *User's Manual* and a *Reference Manual* of several hundred pages each with separate tables of contents and indices. The *User's Manual* is by far the best of the three 386 FORTRAN compilers' considered, clearly documenting the compiler switches, FORTRAN-386 extensions and peculiarities, interfacing with other languages, compatibility with the Unix f77 compiler, optimization, porting issues, intrinsic functions, and numeric coprocessor support. Stephen Fried from MicroWay has written probably the best piece on numerics and optimization I've seen as one chapter of the *User's Manual* (see his earlier *BYTE* articles, too).

FORTRAN-386 sports one feature not available from the other vendors—it figures out which FORTRAN variables must be SAVEd and allocates globals for them. This makes porting from IBM and VAX much easier. The other compiler vendors do allow making all variables SAVEd, but with a noticeable performance penalty. MicroWay is working on improving the numeric code generator. The version I tested thinks that the math chips have registers, as opposed to a stack, and as a result generates much better code for the Weitek chip than it does for the 80387. This can represent up to a 20% speed penalty. For the benchmarks I ran, the compiler generated fast times although all of these compilers generated fast code compared with the real mode DOS compilers. But FORTRAN-386 also needs lots of memory for operation. One benchmark program (Superlite) with one program unit of 2,000 lines ran out of memory during compilation even on a machine with 3 megabytes.

SAIC FORTRAN-386

Science Applications International Corporation (SAIC) has been working with Silicon Valley Software (SVS) of Cupertino, California, to market SVS's FORTRAN, Pascal, and C compilers under DOS. SAIC's Los Altos office needed a 386 FORTRAN compiler for its own use and ultimately decided to offer the same product in the DOS market. SVS traditionally markets their compilers, written in Pascal, to the Unix world. The SVS FORTRAN compiler executes under the IGC DOS extender and generates code for this same runtime. The compiler supports the 80287, 80387, and Weitek chips. SVS FORTRAN-386 comes bundled with the

Table 5. Small Mainframe FORTRAN Program Execution Times

(minutes:seconds)

	LPI	MicroWay	SAIC	Lahey	DOS (1)	Definicon	VAX
CHILC (3 files--1,170 lines) tests double precision math and logical I/O							
Compile time	1:26	2:36	0:31	0:22			
Link time	0:28	0:10	0:10	0:09			
Total	1:54	2:46	0:41	0:31			
OBJ file size	33,955	50,897	29,266	40,393	51,169		
EXE file size	133,066	167,992	67,156	70,144	65,612	162,848	
Execution time	11:56	10:20	died (2)	17:27	15:08	9:39	8:15
SUPERLITE (1 file--2,420 lines) tests transcendentals							
OBJ file size			75,589	99,404	111,986		
EXE file size			454,860	121,988	118,508		
Execution time	(4)	(4)	3:24	7:02	6:16	4:15	1:53
UPDATE (1 file--1,890 lines) tests sequential file I/O (see Note 3)							
OBJ file size	32,991	36,603	27,788	47,608	76,444		
EXE file size	130,826	286,940	212,856	76,040	92,209		
Execution time	3:04 (5)	failed	2:36	2:51	1:36		

NOTES: DOS compilers tested on Compaq 16 MHz 80386/387; using VMS FORTRAN on VAX 11/785+ with FPA, and Definicon 20 MHz 68020/68881 using SVS FORTRAN Compiler.
(1)The best PC DOS execution times are based on Intel FORTRAN-86 for Superlite and Update, and Austec RM FORTRAN 2.42 for Chilc.
(2)An earlier version 2.6 of SVS executed in 13:41
(3)Update was tested with a 160k source deck and a 4k mods file to produce a 450k FORTRAN file and a 250k library file.
(4)Superlite contains one large procedure of over 2,000 lines—both LPI and NDP ran out of memory compiling in a 2 megabyte Compaq 386
(5)Times are running under Phar Lap; under X-AM execution time was 3:43

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Phar Lap linker and is also the only compiler to come bundled with a source level debugger. After seeing the rather primitive nature of the 386 assembly language debugging, SAIC insisted that SVS port their source debugger to the DOS environment. The SVS debugger works only with the SVS compilers since the compiler back end generates the DBG file. IGC added two features to X-AM so that SVS could port its debugger to their environment—the ability for one program (the source debugger) to load another program and the ability to take over the breakpoint interrupt vector.

The SVS compiler supports a variety of command-line and inline (source) options, but no control over optimization; it's always on. The listing options don't provide any assembly language output. A SAVEall option can be used to allocate globals for all variables. Otherwise, SVS FORTRAN will allocate variables on the stack. Even common blocks in the main program are allocated on the stack at entry. Invoking SAVEall exacted a 15% performance decrease on some of the benchmark programs I used. The LPI FORTRAN compiler exhibited similar behavior. This is an example of differences between the 8086 and 80386. In the 86 world, accessing absolute memory locations was faster; on the 386 register, indirect data access is quicker.

The SAIC documentation includes a brief *User's Guide*, the *SVS FORTRAN Reference Manual*, and the *Debugger Reference Manual*. None has an index. The little DOS-specific information is buried in two appendices in the *FORTRAN Reference*. I knew where to look only because I reviewed the SVS 68020 compiler on the Definicon coprocessor for the PC in the last issue of *PJ*.

The SVS compilers generate an intermediate file that goes through two other program steps before becoming an object file suitable for the Phar Lap linker. Despite these steps, this compiler executed the fastest of the three FORTRAN compilers, and the executable file sizes were usually the smallest of the three compilers'. File execution speed was very fast on the 80387, often the fastest of the three compilers'. The source debugger allows setting breakpoints or tracepoints using source line numbers or procedures. The debugger can be used to look at both local and global FORTRAN variables, including those allocated to registers, but it does not provide disassembly of the code generated.

The IGC runtime was awkward to use because the first attempt to execute any sizable FORTRAN program would crash me into the IGC debugger. The culprit was the fixed size stack in combination with the SVS code generator. As noted above, common blocks and variables are allocated on the stack on entry to main at runtime. Each program has to have its IGC runtime profile adjusted to allow an adequate stack for these variables. The next IGC revision should remove this problem.

A.I. Architects' Hummingboard Coprocessor

A.I. Architects manufactures a series of 80386/387 coprocessor boards for the PC XT and AT. These boards provide a separate 80386 CPU and optional math chip and from 2 to 24 megabytes of memory in both 16- and 20MHz versions. The board I used was a 4 megabyte, 20MHz unit that performed flawlessly. This had to be the easiest board I've installed. Four dip switches set the ports used by the coprocessor board. Using the default settings, I dropped the board into an AT&T PC6300 and was running in fewer than two minutes. The Hummingboard (HB) uses a 64K window of memory into the PC or AT address space along with several I/O ports for communication. This memory window can be in one of four locations—9000h, A000h, D000h, or E000h. A CONFIG.386 file specifies the default window address and total memory on the board.

These boards provide an excellent environment for developing and debugging 80386 software. The 80386 CPU can crash on the HB, yet the main PC CPU will still be active to view a post mortem dump or disassembly. The HBs use a 12K memory cache. The coprocessor boards don't have to respond to the PC's timer interrupts, and processing can be performed in parallel on the two CPUs. The designers obviously did something right since the 16MHz HB outperforms a 16MHz COMPAQ 386 machine by 10% or more. (The Compaq/20 is slightly faster than the 20MHz HB.) OS386 running on the HB can also be used to run normal DOS programs although they run blindingly fast. Execution times compared with my base 8MHz AT&T 6300 were about 500% faster on the HB. This board should be considered an alternative to the Definicon 68020 Coprocessor Board reviewed in the last issue of *PJ*.

And the Envelope, Please

Don't pay attention to the 386 executable file sizes—the bloat is mostly from uninitialized data. The compiler writers are revising the way they handle this. As demonstrated by the benchmarks, the 386 protected mode offers benefits to both integer and numeric programming (when using the 80387). C programs requiring the large model or 32-bit ints should show a substantial benefit using the protected mode. These simple benchmarks indicate a decrease in execution times of 30 to 70% of the real mode equivalents. The math chip operates at the same speed in either real or protected mode, but array indexing is improved by the 386. The reason these compiler vendors are rushing to support the Weitek chip for numeric programs is that it promises to double or triple numeric performance over the 80387. MetaWare with High C-386 and Pascal-386 compilers offers an easy entry into the 386 protected mode. Both SAIC and MicroWay will offer C and Pascal compilers, but MetaWare has had a year's headstart, and its libraries are based on several years of experience in the DOS environment.

386|ASM/LINK & 386|DOS Extender
Phar Lap Software, Inc.
60 Aberdeen Avenue
Cambridge, MA 02138
(617) 661-1510

X-AM Development Environment
Intelligent Graphics Corporation
4800 Great America Parkway
Santa Clara, CA 95054
(408) 986-1431

OS/286 & OS/386 Developer's Kit
A.I. Architects, Inc.
One Kendall Square
Cambridge, MA 02139
(617) 577-8052

High C & Professional Pascal
MetaWare Inc.
903 Pacific Avenue, Suite 201
Santa Cruz, CA 95060
(408) 429-6382

LPI FORTRAN, COBOL, & BASIC
Language Processors, Inc.
400-1 Totten Pond Road
Waltham, MA 02154
(617) 890-1155

NDP FORTRAN-386, C, and Pascal
MicroWay
P.O. Box 79
Kingston, MA 02364
(617) 746-7341

SVS FORTRAN, C, Pascal
Science Applications International Corp.
5150 El Camino Real, Suite C-31
Los Altos, CA 94022
(415) 960-5931

This preview of FORTRAN compilers demonstrates the benefits of using any of these products over their real mode counterparts. All three vendors are working hard to improve code generation and adjust to the DOS environment. These vendors need to either support all three DOS extenders or supply their own INIT modules as MetaWare has done. The three DOS extenders are all functional, but more divergent than I like to see. There needs to be some agreement on EXE file formats and a set of uniform calling conventions on a minimal set of DOS calls. Programmers will also need better tools, such as profilers and debuggers that show the 80387 stack and Weitek registers and provide source and symbolic debugging capabilities. ♦

The test files used for this article are available on *PJ*'s bulletin board services and the Issue Disk for Volume 6.2.

M. Steven Baker holds degrees in architecture and electrical engineering from the Massachusetts Institute of Technology and M.A. degrees in architecture and urban and regional planning from the University of Oregon. Steve works as an energy consultant for the Oregon Department of Energy.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Simply the BEST C and Pascal on AT, 386, Sun, Apollo, RT, VAX, 370

"The most rock-solid C compiler in the industry. Superb technical support and portability. Superior code generated."
Gordon Eubanks, Symantec — Q&A (386).

"It simply works, with no trouble, no chasing strange bugs, and excellent warning and error messages ... a professional product."
Robert Lerche, Bay Partners.

"For large-scale software development, the highest quality C compiler available on the market today. Pragmas are great. Quality of support is exceptional." **Randy Neilsen, Ansa—Paradox** (DOS,OS/2).

"15% smaller and 15% faster than Lattice C."
Robert Wenig, Autodesk.

"Our software is running anywhere from 30 to 50% faster than when compiled under Lattice." **David Marcus, Micronetics.**

"We switched from Lattice due to a 10% reduction in code size. The compiler is very stable." **Lee Lorenzen, Ventura Software — Ventura Publisher, marketted by Xerox Corp.**

"Best quality emitted code by any compiler I've encountered. Often amazing." **Bill Ferguson, Fox Software — FoxBase (386).**

"Messages sometimes pointed out type mismatches, incorrect-length argument lists, and uninitialized variables that had been undetected for years [in 4.x bsd]." **Larry Breed, IBM ACIS [RT PC].**

"Diagnostics turned up bugs missed by other compilers. Rapid bug fixes by technical support, someone who knew what he was talking about. 80386 code is well optimized."
Tim Addison, Logistics Data Systems.

"386 protected mode support is fantastic, especially the access to large amounts of memory. It's mainframe compute power on a PC." **Dan Eggleston, Viewlogic.**

"The preprocessor supplied with Professional Pascal is quite useful. The code quality and control over segmentation and memory models are superior to MS Pascal." **Bob Wallace, QuickSoft.**

Check Out These Reviews

- **High C ™:**

Computer Language	February 1986, '87	
Dr. Dobb's Journal	August 1986	
PC Magazine	Jan. 27, 1987	(80386 version)
Dr. Dobb's Journal	July 1987	(80386 version)
BYTE Magazine	November 1987	(80386 version)

- **Professional Pascal ™:**

PC Magazine	Dec. 29, 1985	
Computer Language	May 1986	
PC Tech Journal	July 1986	
Journal of Pascal, Ada, & Modula-2	Nov.-Dec. 1986	
BYTE Magazine	Dec. '86, June '87	(80386 version)

Why MetaWare compilers

- They are specifically designed for serious software developers.
- They are reliable and robust: they don't break at every turn.
- Their generated code is the best, or near best, on each architecture.
- Their superior diagnostic messages help you produce better products more quickly.
- Your source can be ported with ease to the most popular systems.
- You can link mixed-language modules from our compilers, others
- You can benefit from high-level, personal technical support.
- You can take advantage of the latest ANSI C extensions, and/or extensive Pascal extensions. **High C** has been tracking the ANSI Standard for two years; **Professional Pascal** will soon have a VS Pascal compatibility switch and several Apollo Pascal ext'ns.
- You can take advantage of the latest 387 and Weitek 1167 support — we have the only compilers with Weitek real mode support.

MetaWare's High C ™ and Professional Pascal ™

Power Tools for Power Users

Ashton-Tate: dBase III Plus, MultiMate; Autodesk: AUTOCAD, AUTOSKETCH (8087, '387, Weitek); Boeing Computer Services (Sun); CASE Technology (Sun); CAD/CAM giant Daisy ('86, '386, VAX); Deloitte Haskins & Sells; Digital Research: FlexOS; GE; IBM: 4.3/RT, 4680 OS; Lifetree Software (Pascal): Volkswriter Deluxe, GEM-Write; Lugaru: Epsilon; NYU: Ada-Ed cmplr; Semantec: Q&A; Sky Computers; ... (Product names are trademarks of the companies indicated.)

Industrial-Strength

MetaWare C and Pascal compilers are designed for professional software developers. These tools are loaded with options to control them for special purposes. You can adjust the space-time trade-off in code quality. You can adjust external naming conventions to agree with linkers and operating systems. You can specify segment names for segmented architectures, and to help place code or data in particular places for embedded applications. You can select from five memory models for the 8086 family. And on and on.

A Partial List of Optimizations

Common subexpression and dead-code elimination, retention and reuse of register contents, jump-instruction size minimization, tail merging (cross jumping), constant folding, short-circuit evaluation of Boolean expressions, strength reductions, fast procedure calls, automatic mapping of variables to registers (where advantageous), ...

"Platform" — Code Quality

Sun,Apollo,SGI	— 18%, 3%, 26% > resident compiler (Dhrystone).
PC: DOS, OS/2	— 3-10% > Microsoft C; 30% > MS Pascal, LatticeC.
386 32-bit DOS	— no competitors, since November, 1986.
286, 386 UNIX	— 66% better than pcc (Dhrystone, 386).
VAX VMS	— ≈ DEC's excellent C and Pascal; better features.
VAX Ultrix	— 19% > pcc (Dhrystone); much > Berkeley Pascal.
RT PC/4.3bsd	— 89% > the original port of pcc (Dhrystone).
370 CMS,UNIX	— much better than any C, and VS Pascal.
AMD 29000	— >40,000 Dhrystones! Available in Q2, cross.

Since 1979.

MetaWare™ INCORPORATED

903 Pacific Avenue, Suite 201 • Santa Cruz, CA 95060-4429
(408) 429-META (6382) • Telex: 493-0879 META UI

The Clear Choice for Large Programming Projects — PC Tech J.

© 1987 MetaWare Incorporated. MetaWare, High C, Professional Pascal, and DOS Helper are trademarks of MetaWare Incorporated. Others and their owners are duely respected.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

BACK ISSUES

Supplies are Limited—Order Today!

Vol. 1.1: Norton batch files and Pascal I/O; Program style manual; COBOL on the IBM PC; Benchmarks; System analysis; BASIC modules; Rounding error.

Vol. 1.2: Defining character sets; C86 BIOS calls; Troubleshooting; Copy protection; Pascal code; Input routines; Writing an assembler.

Vol. 1.3: ASM dumb terminal; ASM tutorial routines; C source code; ASM graphics line drawing; Text editors reviewed; Utilities; Self-documenting programs; Text files to dBASE II; dBASE II PEEKS & POKES.

Vol. 1.4: Diskette directory via Pascal; MASM WHILEWEND; C/ASM interface; ASM stay resident routine; Structured programming; Software law; Pascal tutorial; Default values and variable names.

Vol. 1.5: Multi-tasking software review; Transporting C programs; Programmer's calculator; dBASE II bargraphs; Utilities; ROM BIOS equates; Pascal check program; Combining modules in Assembler; Structured programming in BASIC; Agreement outline.

Vol. 1.6: ASM printer configuration; DOS patch utility; Hi-speed graphics; Software ownership; Structured programming revisited; C and BASIC compared; Software protection; dWINDOW review.

Vol. 2.1: Freelance consulting; Sorting in C; Pascal text editor source code; CRT distress; Compiled BASIC; Assembly modules for C; Designing a relational database; Ultra Utilities; Punctuation and Style reviewed.

Vol. 2.2: Interview with Brian Kernighan; Understanding C through assembly language; Utility routines in C and assembly; Music in C; Automatic line number optimizing in BASIC; Assembly language tutorial; Full-screen input routines in BASIC; dBASE II printer control codes; Pascal tools reviewed; C programming support products.

Vol. 2.3: Software development environments; Creating a productive screen handling tool; Assembly tutorial; Generating graphics within windows; Composition and editing system for musical ideas; Hybridizing assembly and BASIC; Creating screens; Linking Assembler to Pascal and BASIC.

Vol. 2.4: Writing a custom ROM BIOS for PC DOS; RISC, Transputers, and mass storage; The future of disk protection; 8088 assembly macros; Random access files in BASIC; C utility routines you can buy; Using the PMATE macro language; Choosing a database manager; E/SP syntax-checking; Pascal editor reviewed.

Vol. 2.5: Developing the APX CORE Executive; From assembler to C via a 211 byte .EXE main( ); Scroll control using BASIC and Assembler; Faster screen handling with BASIC and ASM, Random access files, Part 2; RTOOL, a PC-resident programmer's toolkit; dBkey reviewed; Pascal's pointer power.

Vol. 2.6: Periscope--building a hardware board for the IBM PC; Intel 80286 bugs by FNE; AT&T surprises all; Field Usage and pointer subtraction in C; Calendar functions for BASIC; Memory segmentation; Reduce your .EXE file sizes; Enhancing Debug; dBASE II to dBASE III? THINKTANK reviewed.

Vol. 3.1: November's Comdex; C--its uses and abuses; Using the C preprocessor; Memory segmentation examples; Index files, Part 1; Structured programming language; Periscope, a symbolic debugger.

Vol. 3.2: UNIX! (again); Artificial Intelligence; Doctor C's tips--pointers to functions; Calendar date routines in BASIC; Calculate record layouts, automatically!

Vol. 3.3: NAPLPS and the professional programmer; Instant-C; Printer control from a batch file; Repositioning the stars in Starfinder; Reducing EXE file size through data positioning; tools for better programming.

Vol. 3.4: More UNIX stuff; C programming books, Part 1; C code excerpts, Part 1; How to cure the "dreaded locked keyboard" condition; Calling MS DOS from Turbo Pascal; DOS 2.x sort and find filters; the SET Tools.

Vol. 3.5: Windows done here; C programming books, Part 2; C code excerpts, Part 2; A menu-driven PC using DISPLAY.COM; Turbo Pascal and MASM: making them work together; dBASE II routines; Turbo Pascal 3.0; TopView Programmer's Toolkit.

Vol. 3.6: About C's HLL overhead; How to recover all your costs; Report on the Comdex and Usenix shows; The Santa Cruz Operation XENIX for the IBM PC; Common C bugs; CTOOLS2; Turbo Pascal 3.0 and subdirectories; More dBASE II routines; Clipper reviewed; A better way to interface BASIC and Assembler.

Vol. 4.1: FNE on PC/AT hard disk problems and plunging 64K DRAM prices; AUTOINC explained; Sorting using an AVL tree; Combining binary trees; Using DOS to call programs from Turbo Pascal; An operating system for the people; Final installment of Brett Salter's dTOOLS; Two microcomputer graphics books reviewed.

Vol. 4.2: FNE on Intel 80386, single- vs. multi-user, multi-tasking computing, benchmarks; The home office syndrome; Frank Greco's industry news; Funky string functions; Capturing file handles in Turbo Pascal; Proposed ANSI C Standard, Part 1; Optimizing for speed in Assembler; Memory resident control over print screen functions; Structuring BASIC output with field statements; QuickBASIC Compiler and networking; Programming for Printer Output reviewed.

Vol. 4.3: FNEanother RISC machine, the paging game, software viewpoint, programmer talent, and Aloha, 32032; Sun Microsystems' next generation PC; Consulting—reusable modules; INT24.PAS, a critical error handler for Turbo Pascal; Memory model independence in Assembler; An approach to windowing for the assembly language programmer; Proposed ANSI C Standard, Part 2; Microsoft and Lattice C 3.0 reviewed; Accelerator boards for software development reviewed.

Vol. 4.4: Hal W. Hardenbergh on adult toys, tomorrow's micros, HALGOL update, benchmarks, and RISCs; Frank D. Greco on parallel computers, RISC, UNIX; Consulting—efficient custom programming; DOS 2 file handles and file I/O; Programming the WordStar Document File Conversion Utility; More Optimizing for Speed; Teach an old compiler a new trick; Memory resident programs in C; Tips on formatting with BASIC PRINT USING statements; the structure of dBASE II and III file headers.

Vol. 4.5: Hal W. Hardenbergh on 32-bit PCs, the 68040, Weitek's data-path part, etc.; Frank D. Greco on large screen monitors, language translators, UNIX, and more; Prolog intro, Turbo Prolog reviewed; Consultingtax traps, consulting styles, nest eggs; ASM—Search and Display; ASM—the GROUP statement; The proposed ANSI C Standard, Part 3; C—measuring stack usage and Lattice C's startup code; Short boards reviewed.

Vol. 4.6: DTACK GROUNDED; Unix Books, Part 1; How many amoebas at one o'clock; DOS functions 45H and 46H; DOS Terminate ans Stay Resident; Turbo Pascal inline code; Pascal—Assembler souce code for DOS function calls; Alignment problems in C and Managing memory allocation; Proposed ANSI C Language Standard, Part 4; The Canon A1 laser printer; Genifer dBASE applications generator reviewed; Advanced Trace86 debugger reviewed.

Vol. 5.1: DTACK GROUNDED; The new DOS; Marketing and rules to live by; File redirection; Inside the EGA; String allocation in Pascal; ANSI C Language Standard, Part 5; More tips on formatting BASIC data.

Vol. 5.2: DTACK GROUNDED; Starting your own software house; Pink noise; TSR graphics print screen utility; Coding guidelines for resident programs; Parallel processing with the EGA; C—portability in practice and using header files; Proposed ANSI C Language Standard, Part 6; Indexing in BASIC; A first look at Turbo C.

Vol. 5.3: Starting your own software house--publicity; Unix books, Part 2; Consulting—merger mania; EGA data control; Modula 2 for Pascal lovers; Optimizing Turbo Pascal; ANSI C language standard; Coping with ANSI C and forcing file updates; Formatting BASIC output with buffered PRINT USING.

Vol. 5.4: Starting your own software house—technical support; Unix books, Part 3; Consultant productivity and effectiveness; EGA set/reset capabilities; ANSI C language standard; Character tables and string functions in C; MWWIND—a menu with windows utility; Variable passing and hiding in QuickBASIC 2 subprograms.

Vol. 5.5: Intel's 80386 vs. OS/2; Metagraphics President Jack Davis interview; Consulting—making computer tools pay for themselves; ASM—the invisible CapsLock; TSRs using expanded memory with Turbo Pascal; Note on optimizing Turbo Pascal; Proposed ANSI C Language Standard; Printer output from Turbo C programs; Matchup of Turbo Basic 1.0 and QuickBASIC 3.0.

Vol. 5.6: Multics, OS/2, paging and segmentation; The Usenix conference and other news; How to get rich in consulting; Shell in Marshal Pascal; Turbo Pascal 4.0 preview; Handling fixed length character arrays in C; The proposed ANSI C Language Standard.

Vol. 6.1: Consulting is an attitude; Write mode 3 of the VGA; The LIM 4.0 expanded memory specification; Environment variables in Turbo Pascal 4.0; A timer program in C; Porting mainframe software to the PC--FORTRAN compilers under DOS and the Definicon 32-bit coprocessing engines.

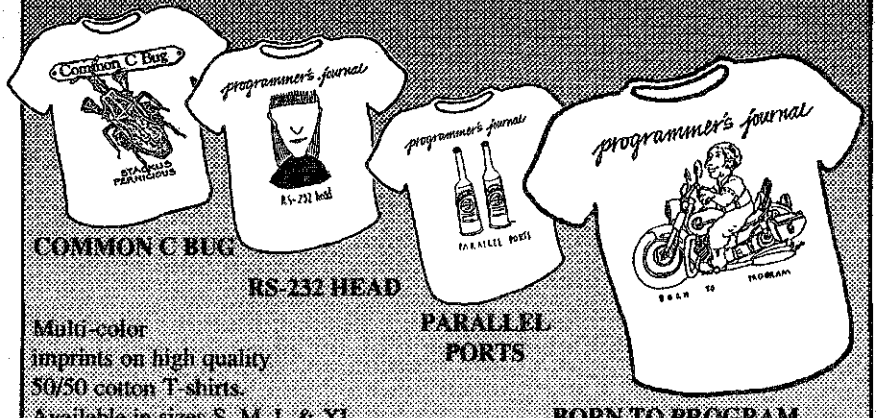
ALL ORDERS MUST BE PREPAID

Use the order form on the card opposite.

Or save time—order by phone using your VISA/MC— CALL: (503) 747-0800

10% discount on orders of 10 or more.

T-SHIRTS



COMMON C BUG

RS-232 HEAD

PARALLEL PORTS

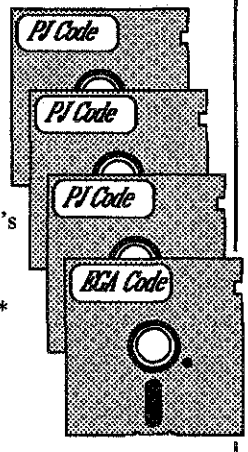
BORN TO PROGRAM

Multi-color imprints on high quality 50/50 cotton T-shirts.

Available in sizes S, M, L & XL

$10.00 each —includes shipping & handling

CODE DISKS



The code from *PJ* Volume 5.5 forward, and from Michael Abrash's EGA series is now available on 360k MS-DOS disks.*

$10.00 each, includes shipping & handling

* Other formats available

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

UNLEASH YOUR 80386!

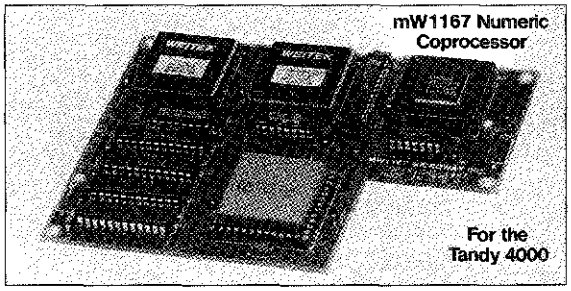
Your 80386-based PC runs at least twice as fast as your old AT. This is good, but not great. The products described below will unleash the true potential of your 80386, giving you 4 to 16 times the power of your old AT. These new MicroWay products include a family of 80386 native code compilers and the mW1167 numeric coprocessor.

Examples of the increases in capacity and performance include:

- Programs compiled with MicroWay NDP Fortran-386 execute 2 to 8 times faster than those compiled with existing 16-bit Fortrans. NDP Fortran-386 can also address up to 4 gigabytes of memory instead of the standard 640 kbytes. MicroWay's NDP compilers and the programs they generate run on MS-DOS or Unix V.
- NDP Fortran-386 generates code for the 80287, 80387 or MicroWay's mW1167. The mW1167 has a floating point throughput exceeding 2.5 megaflops, which is 4 to 5 times the throughput of an 80387 and is comparable to the speed achieved by the VAX 8600.

Equally important, whichever MicroWay product you choose, you can be assured of the same excellent pre- and post-sales support that has made MicroWay the world leader in PC numerics and high performance PC upgrades. For more information, please call the Technical Support Department at

617-746-7341



MicroWay® 80386 Support

MicroWay 80386 Compilers

NDP Fortran-386 and **NDP C-386** are globally optimizing 80386 native code compilers that support a number of Numeric Data Processors, including the 80287, 80387 and mW1167. They generate mainframe quality optimized code and are syntactically and operationally compatible to the Berkeley 4.2 Unix f77 and PCC compilers. MS-DOS specific extensions have been added where necessary to make it easy to port programs written with Microsoft C or Fortran and R/M Fortran.

The compilers are presently available in two formats: Microport Unix 5.3 or MS-DOS as extended by the Phar Lap Tools. MicroWay will port them to other 80386 operating systems such as OS/2 as the need arises and as 80386 versions become available.

The key to addressing more than 640 kbytes is the use of 32-bit integers to address arrays. NDP Fortran-386 generates 32-bit code which executes 3 to 8 times faster than the current generation of 16-bit compilers. There are three elements each of which contributes a factor of 2 to this speed increase: very efficient use of 80386 registers to store 32-bit entities, the use of inline 32-bit arithmetic instead of library calls, and a doubling in the effective utilization of the system data bus.

An example of the benefit of excellent code is a 32-bit matrix multiply. In this benchmark an NDP Fortran-386 program is run against the same program compiled with a 16-bit Fortran. Both programs were run on the same 80386 system. However, the 32-bit code ran 7.5 times faster than the 16-bit code, and 58.5 times faster than the 16-bit code executing on an IBM PC.

NDP FORTRAN-386™ **$595**
NDP C-386™.....**$595**

MicroWay Numerics

The **mW1167™** is a MicroWay designed high speed numeric coprocessor that works with the 80386. It plugs into a 121 pin "Weitek" socket that is actually a super set of the 80387. This socket is available on a number of motherboards and accelerators including the AT&T 6386, Tandy 4000 and MicroWay Number Smasher 386 (Jan. '88). It combines the 64-bit Weitek 1163/64 floating point multiplier/adder with a Weitek/Intel designed "glue chip". The mW1167™ runs at 3.6 MegaWhetstones (compiled with NDP Fortran-386) which is a factor of 16 faster than an AT and 3 to 5 times faster than an 80387 .....**$1495**

Monoputer™ - The INMOS T800-20 Transputer is a 32-bit computer on a chip that features a built-in floating point coprocessor. The T800 can be used to build arbitrarily large parallel processing machines. The Monoputer comes with either the 20 MHz T800 or the T414 (a T800 without the NDP) and includes 2 megabytes of processor memory. Four or more Transputers can be easily linked together to form a Quadputer. A single T800 is comparable in speed with an mW1167-equipped 80386. The compilers to drive one or more Monoputers include Occam, C, Fortran, Pascal and Prolog.

Monoputer T414-20[1] **$1495**
Monoputer T800-20[1] **$1995**
Biputer™ T800/T414[2].....**$4995**
Quadputer™ T414-20[2] **$6995**

[1]Includes Occam [2]Includes TDS

80287 ACCELERATORS

287Turbo-10 .....**$450**
287Turbo-12 .....**$550**
287TurboPlus-12 .....**$629**

80386 Multi-User Solutions

AT8™ - This intelligent serial controller is designed to handle 8 users (16 with two boards) in a Xenix or Unix environment with as little as 3% degradation in speed. It has been tested and approved by Compaq, Intel, NCR, Zenith, and the Department of Defense for use in high performance 80286 and 80386 Xenix or Unix based multi-user systems **$1299**

MicroPort Unix 5.3 is a port of the new Unix 5.3 to the 80386. MicroWay NDP-386 compilers currently run on this version of UNIX.
MicroPort Unix 5.3 .....**from $399**

PC-MOS-386™ is an 80386 operating environment that turns an AT with an AT8 into an MS-DOS multi-user system. The system makes it possible to run applications such as Lotus 1-2-3 on terminals. The operating system also has a Phar Lap compatibility mode that runs programs developed with the Phar Lap versions of MicroWay's compilers .....**from $199**

Phar Lap™ created the first tools that make it possible to develop 80386 applications which run under MS-DOS yet take advantage of the full power of the 80386. These include an 80386 monitor/loader that runs the 80386 in protected linear address mode, an assembler, linker and debugger. These tools are required for the MS-DOS version of the MicroWay NDP Compilers.
Phar Lap Tools.....**$399**

MATH COPROCESSORS

80387-16 16 MHz.....**$495**
80287-10 10 MHz.....**$349**
80287-8 8 MHz **$259**
80287-6 6 MHz **$179**
8087-2 8 MHz **$154**
8087 5 MHz **$99**

MicroWay

The World Leader in PC Numerics

P.O. Box 79, Kingston, Mass. 02364 USA (617) 746-7341
32 High St., Kingston-Upon-Thames, U.K., 01-541-5466

Number 104 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Software Developers

We need your program!

Do you have a program that's good enough to sell, but don't want the problems or financial risk of producing, typesetting, printing, packaging, warehousing, marketing, distributing and supporting a product?

Why start your own software house?
We've done it for you!

Merlin Publishing Group
is now accepting submissions of micro-computer software for publication.

You get:
- initial cash payments
- generous royalties
- to spend your time programming

But we can't help you if you don't submit. Call or write today for our submission guideline kit.

MERLIN
PUBLISHING GROUP ™

1240 Johnsons Ferry Place, Suite A10
Marietta, GA 30068
(404) 977-6034

P. S. See us in Atlanta at COMDEX-Spring '88

Number **141** on the Reader Service Card

Make your C language programs memory resident with
DMS RESIDENT-C

Lattice		Microsoft
"hot-key"		enable
$79.95		$149.95 w/source

Make your assembler programs memory resident with
DMS RESIDENT-ASM

"hot key"		enable
$79.95		$149.95 w/source

American Software International
P.O. Box 523
Windsor, CT 06095-9998
(203) 688-5054

Number **151** on the Reader Service Card

Unbelievable!

SOURCER™

Creates commented source code and listings from memory, .COM files or .EXE files

- **CLARIFY UNDOCUMENTED CODE**
- **EASILY MODIFY PROGRAMS**
- **RECREATE LOST SOURCE CODE**

SOURCER™ creates detailed commented listings and source code directly suitable for assembly. Built in data analyzer and simulator resolves multiple data segments and provides detailed comments on interrupts and subfunctions, I/O ports and much more. Determines all necessary assembler directives. Complete support for 8088 through 80286, V20/V30, 8087 and 80287 instruction sets. No other product comes close to the output quality of SOURCER.

BIOS SOURCE

PS/2 ▪ AT ▪ XT ▪ PC ▪ Clones

- **CHANGE & ADD FEATURES**
- **CLARIFIES BIOS INTERFACES**
- **SPECIFIC TO YOUR MACHINE**

The bios pre-processor to SOURCER™ provides the first means to obtain accurate legal source listings for any bios! Identifies entry points with full explanations. Resolves PS/2's multiple jumps for improved clarity. Provides highly descriptive data labels such as "video_mode" and "keybd_q_head," and much more. Fully automatic.

SOURCER™	$ 99.95*
SOURCER™ with BIOS Pre-Processor	$139.95*

(*OUTSIDE USA, ADD $15 SHIPPING; CA RESIDENTS, ADD SALES TAX)

All our products come with a 30 day money back satisfaction guarantee. Not copy protected. To order or receive additional information just call!

800-538-8157 x 811 (outside Calif.)
800-672-3470 x 811 (inside Calif.)

V COMMUNICATIONS
3031 Tisch Way, Suite 200, Dept. J1
San Jose, CA 95128
(408) 296-4224

PS/2, AT, XT and PC are trademarks of IBM Corp.

Number **125** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Norton Guide for Assembly Language

reviewed by
Michael Abrash

I admit I'm more than a little surprised at just how useful I've found the Norton Guides with the BIOS/DOS Assembly database to be. The concept of on-line reference and/or help software hasn't wowed me in the past, and the thought of sacrificing 65Kb to another TSR seemed out of the question. Too, I like my development environment as it is, and I don't change or add software without a very good reason. Still, as good reports came in, I decided to check out the Norton Guide for Assembly.

I'm glad I did. The Norton Guide has become the first TSR since Sidekick to earn a permanent place in my AUTOEXEC.BAT file. While the Norton Guide doesn't really replace a shelf full of manuals and books, it does provide an appreciable fraction of the information in several manuals—and that information is *very* easy to access.

The heart of the Norton Guides is the Instant Access Program, which is TSR software that provides a common mechanism for searching any database conforming to its format. The Instant Access Program and databases for Assembly, C, BASIC, Pascal, and OS/2 can be purchased separately. If you have the Instant Access Program, you can create your own databases. Norton's databases are tree structures of varying depth that lead ultimately to entries on particular topics. The Norton Guide supports cross-referenced entries and some word searching, including the optional ability to automatically search for the word the cursor is on when the Norton Guide is activated.

That may sound pretty standard, but it works like a charm. I explored the entire Assembly database with ease before I ever read the manual; if I hadn't been reviewing the product, I doubt I'd have bothered with the manual at all. The interface works so intuitively that I spent a good deal of time exploring at random, learning quite a bit in the process. For example, a quick scan of the Assembly database's list of DOS functions showed me all the functions added since DOS 3.0, conveniently marked in a single scrolling list. Consequently, the database has considerable value as a learning tool, all the more so because it costs much less than the set of manuals it summarizes.

Those summarized manuals include IBM's *DOS Technical Reference* and *BIOS Interface Technical Reference* manuals, Microsoft's *Macro Assembler* manual, and Intel's *iAPX 86,88 User's Manual*. The BIOS/DOS Assembly database includes tables and specifications from those documents, along with summaries where appropriate and notes about programming implications and side effects. While the information in the database is not as comprehensive as that in the manuals, much of the important stuff is there and infinitely easier to find. Areas covered in the database include the obvious: the 8088 instruction set; BIOS functions and low-memory usage; DOS functions, FCBs, PSPs, and error handling; and MASM psuedo-ops, operator precedence, and reserved words. Also included are nice touches such as a summary of system ID bytes, keyboard codes returned by the BIOS, an ASCII table, and a diagram of the line-drawing characters actually connected into boxes to facilitate selection.

What all that information boils down to is precisely those annoying bits and pieces that interrupt my programming train of thought and force me to go searching through the (generally inadequate) index of the appropriate manual. The Norton Guide means that I will never again have to search for the extended keyboard code returned by the Gray-Plus key, find the ASCII codes for the four line-drawing corner characters, or remember which registers pass what parameters to the many BIOS video functions, and for that I am eternally grateful.

While the Norton Guide for BIOS/DOS Assembly is certainly handy, it's not perfect. For one thing, it's a bit behind the times since it covers neither MASM 5.0 nor the PS/2 BIOS. Second, while I can't vouch for its thoroughness in every respect, it's good but not great with regard to video—lacking, for example, bit-by-bit information on the EGA BIOS variable at 0:487h. More significant is the lack of hardware documentation, such as I/O ports and hardware interrupts; in certain applications—high-performance graphics programming, for instance—such information is essential.

Overall, I'm convinced that the Norton Guide is well worth the price in both money ($50 for the Instant Access Program, $50 for the BIOS/DOS Assembly database) and memory (65Kb, along with 600Kb+ of disk space) for anyone who's serious about assembly language programming. Actually, it turns out that the Norton Guide has a pretty good solution for the loss of that 65Kb of memory, too. You can invoke the guide so that it loads and runs another program; the guide is available while that program runs, then unloads when the program ends. With a simple batch file or two, the guide can be present in memory only when it's needed, during editing and debugging. In this and many other respects, the Norton Guide for BIOS/DOS Assembler is a well-thought-out product indeed. ♦

The Norton Guides
Peter Norton Computing, Inc.
2210 Wilshire Boulevard, Suite 186
Santa Monica, CA 90403
(800) 451-0303
Fax (213) 453-6398 (technical support), MCI Mail: PNCI
Instant Access Program $50.00, Database (DOS/BIOS Assembly, C, BASIC, Pascal, or OS/2) $50.00

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Program Translation

BAS_C & BAS_PASCAL
Gotoless Conversion
P.O. Box 835910
Richardson, TX 75083
214/404-1404

Gotoless Conversions has released Version 2.6 of BAS_C and BAS_PAS, translators from BASIC to either C or Pascal. The translators come in economy (C $199, Pascal $149) and commercial (C $375, Pascal $280) versions. The economy versions are limited to 1,000 BASIC statements per file, the commercial by available memory. A demo version is available. The translators accept IBM BASICA or similar dialects (Apple BASIC) and generate standard C or Pascal. If the equivalent C or Pascal statement does not exist, the translator converts the BASIC statement into a Pascal procedure or C function call that uses their BASIC Runtime Library. For a limited time, the appropriate Runtime Library is included in the purchase price. Runtime libraries are provided as source code for Turbo Pascal, Microsoft Pascal, Eco-C88, Instant-C, and Lattice, Aztec, Microsoft, and Turbo C. Using the library source code, you can construct any C model.

Options are available via menu or command line to include the BASIC source code as comments, set target indentation and line length, and select source dialect of BASIC (IBM BASICA, QuickBASIC, or Compiler) and target dialect of C (K&R, Lattice, Aztec, and Turbo). The manual covers use of the translators and provides a one page entry for each BASIC statement describing the function, the equivalent C or Pascal syntax, and an example translation from BASIC to the target language. The translators execute nine passes and report statistics on the number of statement types (IF, WHILE, procedures, etc.) for both source and target on completion.

P-tral
Woodchuck Industries
340 West 17th Street, 2B
New York, NY 10011
212/206-6490 or 924-0576

Woodchuck Industries has released Version 1.02 of P-tral, their BASICA/MS-BASIC to Turbo Pascal translator. P-tral has been available for Apple computers since 1985, and the MS-DOS version was released last spring (each sells for $179). P-tral is a three phase processor. The first phase does a syntax check and encodes each statement in an internal language. The second performs static analysis of the BASIC program to determine subroutines and forward references, and to match FOR/NEXT and WHILE/WEND statements. The final stage is interactive, requiring you to rename variables that don't meet Pascal criteria, name subroutines, and dimension dynamic arrays at fixed sizes. BASIC variable type suffixes %, #, and $ are dropped, and duplicate variable names created require new user variable names. New variable names are required for any Pascal reserved words or P-tral standard identifiers. Any P-tral library procedures or functions needed by the generated program are included automatically. Options to control the target file format or include BASIC statements as comments are not available. A well-written manual describes the differences between BASIC and Pascal data types and expressions, other translation concepts, and Pascal program structure. A reference section includes a list of known problems and size limits.

B-Tran
Software Translations
The Carriage House
28 Green Street
Newburyport, MA 01950
617/462-5523

Software Translations has released B-Tran Version 6.10, their QuickBASIC to C translator for DOS. This software from Great Britain is new to the U.S and is an example of the better British software products introduced to the U.S lately. B-Tran translates Microsoft compatible BASIC dialects (MS-BASIC, GW-BASIC, and QuickBASIC) and offers runtime object libraries for Microsoft, Lattice, Turbo, and Quick C. The software includes the translator and one C support library. Microsoft, Lattice, and Turbo C versions cost $449 each and currently include support libraries in two models—Medium and Large. The Quick C version costs $189 and supports the Medium memory model.

B-Tran will actually go through a complete translate, compile, and link process for the target C compiler. B-Tran is invoked with a variety of command-line options to include the BASIC source code as comments, line labels in the executable file for error messages, floating point approximations, special handling of BASIC FOR loops, and float to integer rounding. From the command line you can also force integer multiplication to produce doubles as BASIC does, segment the target C program into modules, etc. B-Tran also supports adding META commands to the BASIC source code to ease translation. META commands are placed after a REM statement, providing a means to note static and dynamic arrays, where to begin segmenting a large BASIC source file, etc. The manual was easily the best among those for the BASIC translators I looked at, covering installation, translator reference use and options, the runtime system, sample translations, translatation error messages, META commands, interfacing to external routines, BASIC command reference, B-Tran limitations, mapping function keys, porting, and string handling. The BASIC reference devotes a page to each command along with an example translation to C. B-Tran is also available under VAX VMS, SCO Xenix, and most variants of Unix.

Turbo-to-C Translator Version 1.39
TGL
27096 Forest Springs
Corvallis, OR 97330
503/745-7476

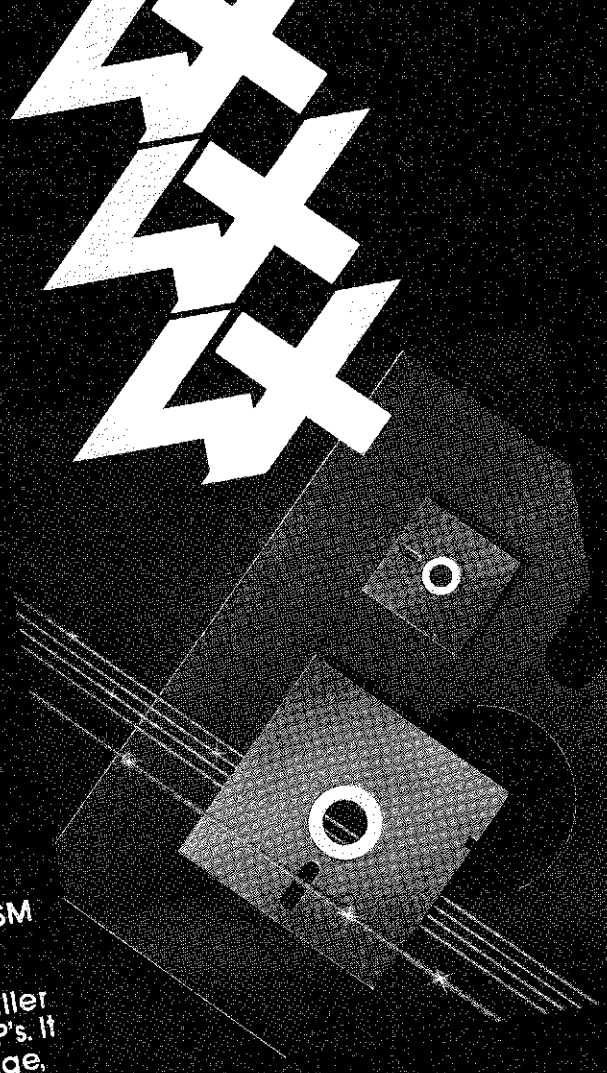
TGL has released Turbo-to-C Translator Version 1.39, their DOS Turbo Pascal to C translator ($495). The translator is a three pass program and currently supports Turbo Pascal 3.0. The first program (MODMAKER) searches the source file for include directives and creates modules containing global declarations. This phase helps in converting monolithic Turbo Pascal programs to a modular C environment. The second program (TBPTOI) preprocesses these modules and produces intermediate files. This step unnests procedures, functions, and identifiers and relocates them to the global declaration section and handles Pascal WITH statements. The last program (TBXLATOR) translates the intermediate files to the target C code. A driver program is supplied to run all three phases of the translator. The three phases may also be used independently. The current translator provides command-line options to support Turbo Pascal window, basic graphic, and screen handling routines. Turtle graphics routines are not currently supported. Other options are a verbose mode, the ability to remove underscores from Pascal identifiers (Pascal generally ignores underscores), conversion of Pascal reals to floats (rather than doubles), line length, indentation, and logical operator and string handling. The package comes with runtime library source code in C to support Pascal functions. The manual is too brief—probably the weakest part of the product. TGL also sells Pascal to C translators for Pascal dialects including UCSD, Pascal MT+, Microsoft, Apple Lisa, Intel, and Apollo Domain. These Pascal to C translators are a bit spendier than the Turbo Pascal 3.0 version—they start at $5,000.

RTC Plus
Cobalt Blue
1683 Milroy, Suite 101
San Jose, CA 95124
408/723-0474

Cobalt Blue has released Version 1.45 of RTC Plus, their FORTRAN and RATFOR to C translator for $450. RTC was originally designed as a RATFOR to C translator. A FORTRAN to RATFOR translator was added to provide FORTRAN to C translation

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

That's right. 4 times faster.



Clocking in at over 75,000 lines per minute on a 6MHz IBM AT, OPTASM is four times faster than Microsoft's MASM 5.0. **4 times faster —** that's 400% more throughput!

But speed is only one part of it. OPTASM is nearly 100% compatible with MASM 5.0 (except 386 support).

It is the only single assembler capable of supporting the various incompatibilities between MASM 3, 4 & 5. That makes OPTASM **more MASM compatible than any single version of MASM!**

Other features? OPTASM generates smaller code without ever generating extra NOP's. It automatically handles jumps out of range, up to 15,000 symbols and most of MASM's phase errors. It also boasts a built in MAKE and simplifies segmentation.

That's why we can make our OPTASM challenge: Test OPTASM head to head against MICROSOFT MASM 5.0. Order **both** assemblers with their 30-day guarantees. In a lot less than 30 days, you'll see just how dazzling OPTASM's speed really is. You'll realize that we're compatible, easier to use, and deliver many more important features than MASM. So accept our challenge. Try both assemblers. **Four times faster and more features, too.** We know which one you'll send back.

Write or call us to order or for our detailed brochure.

WHAT DO PROGRAMMERS SAY ABOUT OPTASM?

"It (OPTASM) just blows MASM away . . . reduces my assemble time for Periscope from 3-plus minutes to less than 45 seconds."
Brett Salter, President,
The Periscope Company

"OPTASM has been absolutely solid. For me, the most useful new product in 1987."
Chris Dunford, Columbia, MD

SLR SYSTEMS

1622 N. Main Street
Butler, PA 16001
412-282-0864
BBS 412-282-2799
Telex 559215
800-833-3061

OPTASM: $195 Guaranteed returnable within 30 days.

Number 165 on the Reader Service Card.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

capability. RTC Plus is primarily a sophisticated macro processor normally run by a batch file to execute various steps. The first stages read various macro files for generic translation macros that apply to most FORTRAN code. These macros rename those FORTRAN names that would conflict with C keywords or library routines. Macro files can be customized with user-written macros to force arguments passed by value and pass unsupported FORTRAN statements through the translator. The translator phase (rtc) makes two passes. The first pass identifies all variables, arrays, functions, and subroutines and detects errors. The second pass generates the C code. After function declarations, statements are translated on a statement by statement basis (i.e., no optimization). RTC comes with C library source code routines for various FORTRAN intrinsic functions and keywords.

The manual is solid, containing tutorial and reference sections. It covers both RATFOR to C and FORTRAN to C translation issues and supported and unsupported statements and data types. RTC does not support FORTRAN I/O statements, FORTRAN-77 character statements and expressions, complex constants and expressions, DATA statements with implied DO loops, COMMON block variables with different names, passing multi-dimensioned arrays (matrices) to subroutines, ENTRY and alternate RETURN, and overlapping EQUIVALENCEs. And the translator aborts code generation if errors are detected. *(to be continued)*

Communications

Portal System
Portal Communications
19720 Auburn Drive
Cupertino, CA 95014
408/973-9111

Portal provides an online communications service for organizations and individuals. The Portal System provides a typical online communications network, including electronic mail, online conferences, computer meetings, and an electronic bulletin board, but in addition, it supports some very attractive features for programmers. At a flat fee of only $10/month for access, with no other charges, Portal also supports direct connection via PC-Pursuit (Telnet's low cost off-peak phone service—$25/month flat fee). Portal provides electronic mail connections to UUCP, ARPA, BITNET, and other mail networks. They also provide connections to other computer networks including Usenet, an international Unix conferencing network. Portal also supports applications, including customer support, user group communications, advertising, and information distribution. Their dialup modem line for access and online subscription is 408/725-0561.

I was most impressed with the mail connections and the gateway to Usenet. Finally, I can communicate easily with university folks on BITNET and ARPANET and with the Unix community connected to Usenet. Usenet is a subset of UUCP-NET, an informal network of Unix sites willing to pass both email and "news" files around the 10,000+ UUCP network. These postings to Usenet consist of about 1.5 megabytes of articles and programs submitted to the network daily. The entire 1.5 megabytes is passed from machine to machine and changes each day. Portal keeps some of the more interesting articles and submissions to Usenet around on their system. In looking through the Usenet gateway, I found a wealth of interesting software (mostly C). Portal's Usenet holdings also provide excellent information on such subjects as TCP/IP and various communication protocols (see below). Definitely an interesting service.

DDN Protocols - IP, TCP, FTP, etc.
DDN Network Information Center
SRI International, Room EJ291
333 Ravenswood Avenue
Menlo Park, CA 94205-3493
800/235-3155 or 415/859-3965

Have you been hearing the acronyms TCP/IP, FTP, SMTP, X.25, and TELNET lately? They stand for various communication protocols standardized by the Defense Department and used primarily on networks. The DDN Network Information Center (NIC) collects and distributes information about the Defense Data Network (DDN) and the ARPANET. Funded by the feds, NIC provides general reference and user assistance to DDN, ARPANET, and Internet users. NIC houses the online reference set of RFCs (Requests for Comment), which are technical notes describing network protocols and technology, and provides these documents as publications either by network access or as hard copies. Many of these protocols have become defacto industry standards for peer-to-peer networking across a variety of machines and hardware. TCP/IP, for example, is now included with most Unix V implementations. TCP/IP consists of Internet Protocol (IP), responsible for internetwork routing and delivery, and TCP (Transmission Control Protocol). Running on top of these protocols are higher level protocols such as SMTP (Simple Mail Transfer Protocol) and FTP (File Transfer Protocol). If it all sounds like alphabet soup to you, publications from NIC might just clear the broth.

NIC distributes a free list of available publications. The RFCs, which describe the technical aspects of these protocols, implementation details, and related background information, are available in hard copy for around $5.00 each. A typical RFC may run 70 pages. NIC also sells for $110 the entire *DDN Protocol Handbook*, a three volume reference set (30 pounds) comprising a large number of previous RFCs. If the handbook sounds too expensive, I would suggest the following RFCs at $5.00 each:

- RFC-791 Internet Protocol (IP)
- RFC-793 Transmission Control Protocol (TCP)
- RFC-821 Simple Mail Transfer Protocol (SMTP)
- RFC-959 File Transfer Protocol (FTP)

All of the RFCs are also available via ARPANET (free) for file transfer from SRI-NIC.ARPA in the netinfo area. You may also request that RFCs be sent to you via electronic mail by addressing SERVICE@SRI-NIC.ARPA with the Subject set to the RFC number you want.

Standards

P1003.1 POSIX Standard
IEEE Standards Office
345 E. 47th Street
New York, NY 10017-2394
212/705-7960

For the last two years, the P1003 IEEE committee has been developing standards for a Portable Operating System Interface for Computer Environments (POSIX)—a standard definition for Unix. Draft 12 of POSIX (300 pages) has been out for final comment since last October. Once it is officially adopted, the standard will be available from the IEEE Sales Office in Piscataway, New Jersey (201/981-1393). In the interim, the current draft is available from the IEEE standards office for $50. P10003.1 standardizes various system features such as process primitives, process environment, files and directories, input and output primitives, device and class specific functions, the C Language Library, system databases, and data interchange format. The committee has been working closely with the ANSI C committee. About the same time that these activities started, AT&T brought out the Unix System V Interface Definition. One Draft 12 Appendix is a comparison of System VID with the POSIX standard, noting any differences. Other standards are in various stages of development, including P1003.2 (Shell and Utility Application Interface), P1003.3 (Test Methods for Measurement of Conformance to POSIX), P1003.4 (Real Time POSIX extensions), and others.

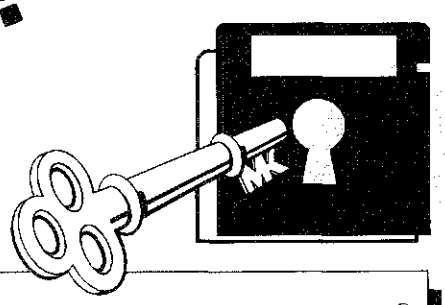
Software Lib

Free Software Foundation
675 Massachusetts Avenue
Cambridge, MA 02139
617/876-3296

The Free Software Foundation (FSF) is dedicated to eliminating restrictions on copying, redistributing, understanding, and modifying software. The "free" means free to change a program (i.e., revise the source code) and redistribute it to a neighbor. Formed by Richard Stallman (of EMACS and MIT AI fame), this group of talented volunteers is developing free compatible replacements for proprietary software. Specifically, they are putting GNU together, a complete, integrated software system for 32-bit microprocessors that is upwardly compatible with Unix. When GNU is released, everyone will be permitted

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MASTER*KEY Unlocks Everything!



turn this into this!

```
C:>DEBUG PROGRAM.COM
-D100 136
8848:0100  EB 18 49 6E 63 6F 72 72-65 63 74 20 44 4F 53 20  k.Incorrect DOS
8848:0110  76 65 72 73 69 6F 6E 0D-0A 24 50 B4 30 CD 21 86  version..$P40N!.
8848:0120  E0 3D 36 01 72 05 3D 0A-02 76 09 BA 02 01 B4 09  `=6.r.=..v.:..4.
8848:0130  CD 21 CD 20 58 EB 2F                             M!N Xk/
-Q
```

MASTER*KEY
No Other Product Comes Close!

An EXPERT may not know the solution, but always knows where to find it.

MASTER*KEY HELPS ANYONE solve those confusing and frustrating software puzzles more rapidly and easily than any other software available, at any cost! It gives you know-how within hours that may otherwise take years of experience. Create a new program from an old one. DON'T REINVENT *THE WHEEL!*

MASTER*KEY - Smart!

MASTER*KEY is an intelligent self-documenting MS-DOS reverse assembler. Its sophisticated procedures swiftly race through massive and baffling object code files to effortlessly discover potential trouble spots.

MASTER*KEY - Educational!

YOU DON'T NEED TO KNOW ASSEMBLY LANGUAGE! MASTER*KEY will take any program from your IBM-compatible computer and return fully-documented, easily-understood assembly language source code (Microsoft MASM 5.0 compatible).

MASTER*KEY - Easy To Use!

MASTER*KEY works both automatically from the DOS command line or interactively from menus similar to Lotus Corporation's 1-2-3 or Symphony. No need to remember any new commands or continually refer to a manual. Use it immediately!

Minimum System Requirements:

256K + 8088/8086/80186/80286/80386 PC
MS-DOS or PC-DOS 2.0 +
One 360K DSDD Floppy Drive (IBM PC Format)

MS-DOS is a trademark of Microsoft.
PC-DOS is a trademark of IBM.

```
H00100: JMP     Short H0011A                    ;00100 EB18          __
;-----------------------------------------------
        DB      "Incorrect DOS version"         ;00102 496E636F727265
        DB      0Dh                             ;00117
        DB      0Ah                             ;00118
        DB      "$"                             ;00119 24
;-----------------------------------------------
H0011A: PUSH    AX                              ;0011A 50            P
        MOV     AH,30h                          ;0011B B430          _0
        INT     21h             ;1-DOS_Ver_Number;0011D CD21         _!
        XCHG    AH,AL                           ;0011F 86E0          __
        CMP     AX,0136h                        ;00121 3D3601        =6_
        JB      H0012B          ; . . . . . .   ;00124 7205          r_
        CMP     AX,020Ah                        ;00126 3D0A02        =__
        JBE     H00134          ; . . . . . .   ;00129 7609          v_
H0012B: MOV     DX,0102h                        ;0012B BA0201        ___
        MOV     AH,09h                          ;0012E B409          __
        INT     21h             ;1-Display_String;00130 CD21         _!
        INT     20h             ;TERM_normally:20h;00132 CD20        _
;-----------------------------------------------
H00134: POP     AX                              ;00134 58            X
        JMP     Short H00166                    ;00135 EB2F          _/
;-----------------------------------------------


MASTER*KEY XREF - PROGRAM.XRF                                  Page 1

0102h                   :   121   2F5   301   320
020Ah                   :   126
03CBh                   :   12B
1-Display_String        :   130   591   610
1-DOS_Ver_Number        :   11D
H00100                  :   100
H0011A                  :   100   11A
H0012B                  :   124   12B
H00134                  :   129   134
H00166                  :   135
TERM_normally:20h       :   132
```

NOTE: The cross-reference is by memory location within the program file!

NOTE: The output is totally Microsoft MASM-compatible.

(not copy protected)

MASTER*KEY will guide you step by step to:

1. Help you learn assembly language, if you desire.
2. Discover how any program runs or why it doesn't.
3. Alter or remove unwanted object code from any program.
4. Incorporate routines from compiled programs into other assembly language, Basic, C, or Pascal programs.
5. Make software more compatible with your computer. Be certain a questionable program won't damage your system BEFORE you run it.
6. Modify software to operate with other versions of DOS.
7. Customize your COMMAND.COM or other executable program directly or by reassembling your altered MASTER*KEY source code.

Order Now! Just $79.95

Phone orders accepted on MC or VISA
$82.45 (includes $2.50 shipping)
$87.65 in California (includes tax & shipping) **C.O.D.** orders add $2.00

(714) 596-0070

Please send **MASTER*KEY!**

Send checks to:
Sharpe Systems Corporation
2320 E Street, Dept. 44, La Verne, CA 91750

Name _____

Address _____

City _____ State _____ Zip _____

Dealer/Distributor Inquiries Welcome

Sharpe Systems Corporation
2320 E Street, Dept. 1610, La Verne, CA 91750 714-596-0070

MASTER*KEY should not be confused with any public domain or share ware software that may have a similar name or be a similar product.

Number 175 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

to copy and distribute it to others. It's an ambitious project. Judging by results so far, they're likely to succeed (though they won't endear themselves to the phone company and Big Blue).

GNU software already available includes GNU Emacs (the premier text editor), GDB (a source-level C debugger), Bison (a replacement for YACC, the Unix compiler-compiler), X Windows System (Version 10.4 by MIT and DEC), MIT Scheme (a dialect of LISP), GNU Chess, and Hack (a display oriented adventure game). After it's uncompressed, this primary GNU distribution is 35 megabytes. FSF also distributes a tape with Beta test versions of GCC (portable optimizing C compiler), GAS (GNU assembler), and the latest version of X Windows (Version 11). The C compiler and assembler currently support VAX and 68000 and 68020 CPUs, and nearby University folks say good things of the VAX GCC compiler. A number of smaller utility programs have also been completed—GNU ls, grep, make, ld, and tar. Emacs is already being distributed with Berkeley 4.3 Unix and on DEC VAXes. This is a lot of source code—Emacs is 7 megabytes, GCC is 3 megabytes, GAS is about a megabyte, and X W indows is over 20 megabytes.

The easiest way to get a copy of GNU software is to ask someone who has it. I will put the smaller utility programs on the *PJ* bulletin board. GNU software is available if you have access to Internet from host prep.ai.mit.edu (the info is in the file /u2/emacs/GETTING.GNU.SOFTWARE). FSF also distributes the software on 9 track, 1600 bpi magnetic tape in Unix TAR format for $150 a tape and sells typeset copies of some of the manuals as a fundraiser. They are always looking for donations of time or computer equipment as well.

Books

The MS-DOS Encyclopedia
Microsoft Press
16011 N.E. 36th Way
Bellevue, WA 98073
800/426-9400

Microsoft Press has released the long awaited and handsomely presented *MS-DOS Encyclopedia* ($134.95). This heady tome (over 1500 pages) is supposed to be the "ultimate resource" for DOS programmers, covering DOS releases through Version 3.2, with an appendix on Version 3.3. The encyclopedia contains version-specific technical data and descriptions for system functions, user commands, and Microsoft programming utilities. Ray Duncan, who wrote the best-selling *Advanced MS-DOS* (also from Microsoft Press) and *PJ*'s highly regarded "Starting Your Own Software House" series, was the encyclopedia's general editor. The encyclopedia is divided into six major sections: (1) a brief history of the development of DOS, (2) a large collection of articles on programming in the DOS environment, (3) user commands, (4) DOS and Microsoft programming utilities, (5) system calls, and (6) 16 appendices on various DOS and PC issues. The articles include some serious source code (also available on disk with the encyclopedia), walk through writing device drivers and TSR programs, and cover interrupt handlers and a number of other issues. The TSR article revealed and documented some DOS features I never fully understood before. The coverage of DOS user commands and utilities is very good.

A remaining anomaly is the incomplete coverage of "undocumented" DOS functions. Some of these undocumented functions are described clearly in the articles, yet omitted in the system function section. I personally think there should be no undocumented calls at all. In addition, even a mention of Netbios is conspicuously absent.

Over a year ago Microsoft Press released the first edition of the encyclopedia and quickly recalled it. Fortunately, I'd bought a copy before the recall. The original book was clearly flawed, with lots of errors, and not very well put together, and it was to their credit that Microsoft made the costly decision to recall it. This time they've done the encyclopedia mostly right. But the flawed, recalled book contained one feature you won't find in the new encyclopedia—multi-page flow charts of the DOS functions and the command loading sequence. At the time I wondered who'd let those charts slip in (a simplified chart for writing a DOS emulator) and speculated, "I bet they won't be in the followup." I was right. ♦

TASKVIEW

ROCK SOLID MULTITASKING!

Packed with the power you need, TASKVIEW takes you beyond the limits of DOS.

- Communicate while you edit
- Compile while you print
- Load up to 5 megabytes of programs
- Manage resident utilities
- Time-slice multiple jobs
- Cut & paste between programs

TASKVIEW lets you load up to 10 of your favorite applications and switch between them at a keystroke. They can even continue to run while you work on something else! TASKVIEW is the BEST multitasker you can buy. Just ask our customers.

"Thanks for a wonderful product!"

"TASKVIEW is the only multitasker I own which runs efficiently & correctly both on my Zenith Z-151 . . . and on my Orchid 286e . . . I also have DoubleDOS, DESQview, TopView, Windows, and Concurrent PC-DOS".

"Fantastic! What DoubleDOS, MS-Windows, and all the other 'stuff' should have been!"

"My BBS is now up 24 hours, 7 days a week thanks to TASKVIEW."

"Thanks! I needed that!"

TASKVIEW requires an IBM PC, XT, AT or Jr compatible, and PC or MS DOS 2.x to 3.x. To get your copy, call toll free:
(800) 367-0651
or send $79.95 + $3.00 S&H ($8.00 Intl.) to:

Sunny Hill Software
PO Box 55278, Seattle, WA 98155-5278

For more information call (206) 367-0650

DoubleDOS trademark Softlogic Solutions, Concurrent PC-DOS reg. trademark Digital Research Corp., MS Windows trademark Microsoft Corp., DESQview trademark Quarterdeck Office Systems, Inc. TopView trademark IBM Corp.

Number 85 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

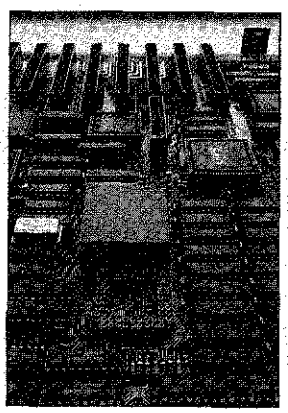
SAVE YOUR PC FROM EARLY RETIREMENT.

GET HAUPPAUGE'S NEW 386 MOTHERBOARD.

386 SPEED—ONLY $1,495

Give your PC a new lease on life! With our industry first 386 MotherBoard, your PC, PC/XT or compatible will revel in speeds equal to the Compaq DeskPRO 386. And faster. Because we've built in 1 Megabyte of high speed RAM and a 387 math coprocessor socket for speeds that will knock you off your rocker.

To keep retirement at bay, our 386 MotherBoard is compatible with the PC/AT (BIOS and I/O) — allowing you to run the new generation of DOS, OS/2. We've also included a 16-bit expansion slot that accommodates the latest I/O expansion card. No accelerator card can give you so much versatility.



With 386 power and true AT software compatibility, your business, desktop publishing and engineering applications will get a boost to boast about! **Technical Features** ▪ 16 MHz 80386 ▪ 1 Megabyte of 100 nsec 4-way interleaved RAM ▪ PC/AT compatible I/O and BIOS for support of OS/2 ▪ Seven 8-bit expansion slots ▪ Two 16-bit expansion slots ▪ One 32-bit RAM expansion slot ▪ Optional 16 MHz 80387 math coprocessor ($695)

Put the power of the 386 into your IBM PC for 1/4 the cost of a 386 computer. And put off your PC's retirement. For more information on our easy-to-install Motherboard, call **1 (800) 443-6284.** In New York, call **(516) 360-3827.**

Hauppauge Computer Works, Inc.
358 Veterans Memorial Highway,
Commack, New York 11725

Hauppauge!

Trademarks: IBM PC, [illegible] Compaq computers

Number **122** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LOW COST ADVERTISING

If you sell products or services to professional programmers, a ***PJ* Billboard** ad is an easy, low cost way to reach more potential buyers. **PJ Billboard** ads must be prepaid and are non-commissionable. Call the ad department for details.

Programmer's Journal
P.O. Box 30160
Eugene, OR 97403
503-747-0800

TURBO PASCAL PROGRAMMERS

The SOURCE CODE for TURBO Pascal is now available. For a remarkably low price users can now have fully commented source code for TURBO Pascal v3.01A (binary arithmetic). This includes the run-time library, editor and compiler. Everything you need is included, even an assembler! Free information.

Price: IBM-PC $67.50, Z-80 $45 (specify disk required)

C.C. Software
1907 Alvarado Ave. #B
Walnut Creek, CA 94596
415-939-8153 **RS# 132**

DRUMA FORTH-83 SYSTEM LANGUAGE

Strict **'83 STANDARD.** Developed for in house use. No 64K limitation. 64K speed & compactness to 320K.

* **IBM PC/XT/AT** & compatibles, **DOS 2.0+**
* **ROM:** Headerless code, separated variables
* Editor, assembler, file & DOS interfaces
* Many powerful and innovative features
* Full Pkg. **$79**, Demo **$10**, S&H **$2**, **VISA/MC**

Druma Forth-83, Druma Inc.
P.O. Box 610097
Austin, TX 78761
512-323-0403 Orders
521-323-2402 BBoard **RS# 172**

FST MODULA-2

Fast two pass compiler with integrated editor and make, large memory model, inline assembler. Requires PC with 512K minimum. Support via BBS and BIX ('fst' conference).

Shareware. Evaluation copy $12, registered version (includes library source code) $39.

PMI's Repertoire® available.

Fitted Software Tools
P.O. Box 956129
Duluth, GA 30136
BBS 404-497-0931 **RS# 138**

PC C SHELL

A faithful implementation of the Berkeley Unix C Shell for MS- DOS. All commands and features applicable to MS-DOS are implemented. A command line editor is built in. Previous commands are recalled using cursor keys. Can be used as the default command processor under DOS 3.X or later. Compatible with all PC software—TSR's, spoolers, etc. RUNS ON ANY MS-DOS COMPUTER!

Price: $50.00, postage paid. Site and source licensing available.

Omega MicroConsulting
722 Rundell Street
Iowa City, IA 52240
319-338-6053 **RS# 183**

PJ CODE IS AVAILABLE!

On disk from:

Programmer's Journal
503-747-0800
See ad on page 76.

On BBS:

FIDO/OPUS Network
Blazer Net 152 Node 9
503-343-9724

Mnematics Videotex
P. O. Box 19
Sparkhill, NY 10976-0019
800-322-3633
914-761-4700 in NY

Turbo Source Search
P.O. Box 876
Scituate, MA 02066
617-545-9131

Microsoft®
QuickC™ / C5.0
Graphics Library
NOW, A LINK!
Hercules™
InColor Card
Graphics Card Plus
Graphics Card
$34.95 Postpaid
Soft Evolutions, Inc.
484 Lake Park Ave., #17
Oakland, CA 94610-2730
(415) 523-8247
Number 177 on the Reader Service Card

ERRATA

Our tables in Steve Baker's "Porting Mainframe Software to the PC" (*PJ*, Volume 6.1, January/February 1988) got a little out of line, so we're reprinting Table 4 from that article.

Table 4. Small Mainframe FORTRAN Program Execution Times (minutes:seconds)

	Austec	DRI	Intel	Lahey	Microsoft	Prospero	WATCOM	VAX
SUPERLITE								
Compile time	failed	failed	13:17	4:23	failed	failed	failed	
Link time			2:13	1:00				
Total			15:30	5:23				
OBJ file size			111,986	99,404				
EXE file size			118,508	121,988				
Execution time			20:47	22:01				1:53
CHILC (3 files)								
Compile time	5:38	failed	5:47	1:17	5:51	3:11	-	
Link time	0:37		1:09	0:35	0:41	0:59	-	
Total	6:15		6:56	1:52	6:32	4:10	0:22	
OBJ file size	51,169		59,249	40,393	51,723	48,550		
EXE file size	65,612		83,766	70,144	60,229	56,576	162,848	
Execution time	50:21		73:14	57:51	61:46 bad output	138:34	Crashed	8:15
UPDATE (Note 1)								
Compile time	7:14	failed	5:57	1:30	6:28	2:59	-	
Link time	0:44		1:59	0:42	0:39	0:48	-	
Total	7:58		8:06	2:12	7:07	3:48	1:04	
OBJ file size	69,256		76,444	47,608	55,173	52,400	-	
EXE file size	68,342		92,209	76,040	60,789	51,200	312,656	
Execution time	14:53		6:12	11:39	7:18	Crashed	Crashed	

1 Update was tested with a 160k source deck and a 4k mods file to produce a 450k FORTRAN file and a 250k library file.

The following corrections to Table 1 regarding Prospero's DOS FORTRAN compiler should be noted: Prospero's compiler supports flagging non-ANSI features, inline comments, mixed language programming (Pascal and assembly), global variables to 31 characters, the underscore in variable names, ANDing numbers, and getting the command line.

Prospero has also released new Pascal and FORTRAN compilers for both MS-DOS and the ATARI that directly support GEM. These are interesting products that come with both clear documentation and built-in compiler support for programming in the GEM graphics environment. ♦

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C CODE FOR THE PC

source code, of course

C Source Code

Bluestreak Plus Communications (two ports, programmer's interface, terminal emulation)	$400
CQL Query System (SQL retrievals plus windows)	$325
GraphiC 4.1 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	$325
Barcode Generator (specify Code 39 (alphanumeric), Interleaved 2 of 5 (numeric), or UPC)	$300
Greenleaf Data Windows (windows, menus, data entry, interactive form design)	$295
Vitamin C (MacWindows)	$200
resident C (TSRify C programs, DOS shared libraries)	$165
Essential C Utility Library (400 useful C functions)	$160
Essential Communications Library (C functions for RS-232-based communication systems)	$160
Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF)	$150
Greenleaf Functions (296 useful C functions, all DOS services)	$150
OS/88 (U**x-like O/S, many tools, cross-development from MS-DOS)	$150
Turbo G Graphics Library (all popular adapters, hidden line removal)	$135
CBTree (B+tree ISAM driver, multiple variable-length keys)	$115
MultiDOS Plus (DOS-based multitasking, intertask messaging, semaphores)	$115
PC/IP (CMU/MIT TCP/IP implementation for PCs)	$100
B-Tree Library & ISAM Driver (file system utilities by Softfocus)	$100
The Profiler (program execution profile tool)	$100
Entelekon C Function Library (screen, graphics, keyboard, string, printer, etc.)	$100
Entelekon Power Windows (menus, overlays, messages, alarms, file handling, etc.)	$100
Wendin O/S Construction Kit or PCNX, PCVMS O/S Shells	$95
QC88 C Compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	$90
JATE Async Terminal Emulator (includes file transfer and menu subsystem)	$80
MultiDOS Plus (DOS-based multitasking, intertask messaging, semaphores)	$80
ME (programmer's editor with C-like macro language by Magma Software)	$75
WKS Library (C program interface to Lotus 1-2-3 program & files)	$65
Quincy (interactive C interpreter)	$60
EZ_ASM (assembly language macros bridging C and MASM)	$60
PTree (parse tree management)	$60
HELP (pop-up help system builder)	$50
Multi-User BBS (chat, mail, menus, sysop displays; uses Galacticomm modem card)	$50
Heap Expander (dynamic memory manager for expanded memory)	$50
Make (macros, all languages, built-in rules)	$50
Vector-to-Raster Conversion (stroke letters & Tektronix 4010 codes to bitmaps)	$50
Coder's Prolog (inference engine for use with C programs)	$45
C-Help (pop-up help for C programmers ... add your own notes)	$40
Biggerstaff's System Tools (multi-tasking window manager kit)	$40
CLIPS (rule-based expert system generator, Version 4.0)	$35
TELE Kernel (Ken Berry's multi-tasking kernel)	$30
TELE Windows (Ken Berry's window package)	$30
Clisp (Lisp interpreter with extensive internals documentation)	$30
Translate Rules to C (YACC-like function generator for rule-based systems)	$30
6-Pack of Editors (six public domain editors for use, study & hacking)	$30
ICON (string and list processing language, Version 6 and update)	$25
LEX (lexical analyzer generator)	$25
Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor)	$25
AutoTrace (program tracer and memory trasher catcher)	$25
C Compiler Torture Test (checks a C compiler against K & R)	$20
Benchmark Package (C compiler, PC hardware, and Unix system)	$20
TN3270 (remote login to IBM VM/CMS as a 3270 terminal on a 3274 controller)	$20
A68 (68000 cross-assembler)	$20
List-Pac (C functions for lists, stacks, and queues)	$20
XLT Macro Processor (general purpose text translator)	$20
C Tools (exception macros, wc, pp, roff, grep, printf, hash, declare, banner, Pascal-to-C)	$15

Data

WordCruncher (text retrieval & document analysis program)	$275
DNA Sequences (GenBank 48.0 of 10,913 sequences with fast similarity search program)	$150
Protein Sequences (5,415 sequences, 1,302,966 residuals, with similarity search program)	$60
Webster's Second Dictionary (234,932 words)	$60
U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	$35
The World Digitized (100,000 longitude/latitude of world country boundaries)	$30
KST Fonts (13,200 characters in 139 mixed fonts: specify TEX or bitmap format)	$30
USNO Floppy Almanac (high-precision moon, sun, planet & star positions)	$20
NBS Hershey Fonts (1,377 stroke characters in 14 fonts)	$15
U. S. Map (15,701 points of state boundaries)	$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409 USA
acwinfo@uunet.uu.net
Voice: (512) 258-0785
BBS: (512) 258-8831
FidoNet: 1:382/12

Free surface shipping on prepaid orders

MasterCard/VISA

Number 30 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ADVERTISER INDEX

Reader Service#	Advertiser Name	Page Number
180	A.I. Architects, Inc.	71
184	American Cybernetics	19
151	American Software International	78
162	Anji Software, Inc.	58
170	ASCII	25
86	Aspen Systems	55
178	Attachmate Corporation	28
30	Austin Code Works	87
51	Az-Tech Software, Inc.	63
2	Blaise Computing	3
3	Borland International	C4
132	C. C. Software	86
*	*C Gazette, The*	8
38	C Ware Corporation	53
56	CompuView Products, Inc.	12
*	Creative Programming Consultants	55
5	Dair Computer Systems	54
128	DOS Users Group	31
172	Druma, Inc.	86
70	E.C.I.	23
8	Ecosoft, Inc.	5
173	Elan Computer Group, Inc.	60
113	Elfsoft	33
138	Fitted Software Tools	86
118	Fleming Software	57
135	Gilmore Systems	24
*	Gimpel Software	59
122	Hauppauge Computer Works, Inc.	85
154	HavenTree Software LTD	11
169	IGC	1
58	Interface Group, Inc.	13
155	Komputerwerk	31
63	Laboratory Microsystems, Inc.	63
44	Lahey Computer Systems	35
141	Merlin Publishing Group	78
143	Metagraphics Software Corporation	16
45	MetaWare, Inc.	75
168	Michael J. Young Software Engineering	53
104	MicroWay	77
179	Mightysoft, Inc.	29
*	PLEASE CONTACT THESE ADVERTISERS DIRECTLY	

Reader Service#	Advertiser Name	Page Number
121	MISOSYS Inc.	17
94	MMC AD Systems	49
109	NanoSoft	33
160	Nu-mega Technologies	67
183	Omega MicroConsulting	86
16	Opt-Tech Data Processing	35
145	Pathfinder Associates	61
71	Pecan Software Systems	17
*	Programmer's Journal Mail Order	76
6	Periscope Company, The	18
146	Programmer's Connection	7
97	Qualstar Corporation	35
147	Quarterdeck Office Systems	C2
46	Quilt Computing	41
*	Raima Corporation	49
156	SAIC	31
174	Seidl Computer Engineering	30
148	SemWare	16
161	Serengeti Software	23
175	Sharpe Systems Corporation	83
165	SLR Systems	81
177	Soft Evolutions, Inc.	86
98	Software Family, The	52
110	Softway, Inc.	57
47	Solution Systems	46
176	Sophco, Inc.	37
134	Stau Enterprises Inc.	23
158	Summit Information Systems, Inc.	62
85	Sunny Hill Software	84
74	Sydetech, Inc.	54
171	Systemat Corporation	10
163	Tenon Software	55
166	Tool Makers, The	49
25	TurboPower Software	33
35	Turbo Source Search	43
125	V-Communications	78
181	Watcom Products, Inc.	44-45
73	Whitewater Group, The	9
123	Xenosoft	63
182	Zortech, Inc.	C3
*	PLEASE CONTACT THESE ADVERTISERS DIRECTLY	

When contacting any of the above advertisers, please tell them you saw their ad in ***Programmer's Journal***!

Come see us at the 13th West Coast Computer Faire. We'll be in booth #439

Articles Coming in the May/June Issue, Volume 6.3

Consulting:
Getting Your Consulting Tools in Order
by Paul Barkley

On Graphics:
Reading VGA Memory ***by Michael Abrash***

Windowing:
Frank Greco on MS Windows, X Windows, NeWS, etc.

BASIC:
Comparing Call, Gosub, and Multi-line Functions
by Ethan Winer

ADVERTISE IN

Programmer's Journal
May/June Issue, Volume 6.3
New Space Reservation Deadline:
March 11, 1988
Materials Due:
March 25, 1988
Call Programmer's Journal Ad Department to reserve your space today!

(503) 747-0800

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

TOOLKITS FOR TURBO C & QUICK C from ZORTECH INC.

HOTKEY

A complete set of Terminate Stay Resident (TSR) functions that help you to write reliable 'pop-up' programs.

Now you can make your programs 'Sidekickable'. Two example programs are included, a 'pop-up Calculator' and a pop-up 'Critical Error Handler'.

The Hotkey toolkit handles all floating point functions in resident mode.

The 32 page manual includes an interesting discussion of the origin and history of undocumented MS-DOS function calls, together with a full explanation of the theory and practical use of TSR's.

Only $49.95! State Turbo C or Quick C version.)

COMMS

Do you need to incorporate serial communications into your applications? Yes! Then get this inexpensive but highly professional COMMS toolkit from Zortech Inc.

Look at the list of features: Xmodem, Kermit and ASCII file transfer, Hayes modem control, VT52, VT100 and ANSI terminal emulation, supports up to 8 serial ports, speeds up to 19.2k baud rate and higher.

Two demonstration programs are included, MINICOM and MAXICOM (like Procomm) together with the 120 page manual and full source code FREE!

Only $49.95! (State Turbo C or Quick C version.)

GAMES

Have you ever wondered how to write a chess program? Now we reveal the secret algorithms and techniques of the masters with this dynamic Games toolkit.

The package comes complete with the full source code to three ready to play games of strategy – Chess, Backgammon and Wari (an ancient African game).

A comprehensive 150 page manual is provided giving an in depth look at the history, structure and program design of such 'Strategy Games'.

Only $49.95!

(State Turbo C or Quick C version.)

SUPERTEXT

This is not simply an 'Editor' toolkit, but a full-blown, 'WordStar' compatible wordprocessor with the full source code.

As well as all the normal editing functions, you will also find 'dot' commands and full printer control. The SuperText toolkit handles files of any size and allows full on-screen configuration.

Do you need to incorporate a wordprocessor into your application? Yes! Then get the SuperText toolkit complete with full source code and 150 page manual now!

Only $49.95! (State Turbo C or Quick C version.)

PROSCREEN

Generate high quality data entry screens with the Pro-Screen – Screen Designer and Code Generator.

You can draw the data entry screen, define the input fields, define the input criteria, set screen colors and attributes, draw single or double lines, make boxes – press a few buttons and 'hey presto' Pro-Screen generates the C source code for your application!

Professional applications programmers will find this versatile utility and it's associated functions invaluable.

Comes complete with a substantial 78 page manual and demo programs.

Only $49.95! (State Turbo C or Quick C version.)

WINDOWS

Add super-fast text screen handling to your applications with the WINDOWS library from Zortech Inc.

Give your applications the professional look – with instant zooming and exploding windows. Incorporate drop-down menus and Lotus style menus with our easy to use functions.

Automatically handles memory saving and buffering of window text. Use any number of overlapping windows in your applications. Write to any window, read from any window, close any window, pull any window to the top.

Over 55 functions together with a big 85 page manual and remember, you get the full source code.

Only $49.95! (State Turbo C or Quick C version.)

NEW! C VIDEO

- *Now learn C the easy way!*
- *Get the 'Complete C Video Course' from Zortech Inc. together with our big 365 page workbook.*
- *Ten 1 hour tapes – 36 lessons!*
- *Easy to follow course, you get an excellent introduction to the C language.*
- *Takes you step-by-step up to the intermediate and advanced levels.*
- *Teach yourself at home or the office – at your own speed.*

only ~~$295.00~~ **$199.95**

Yes! Rush me these items!

☐ HOTKEY
☐ COMMS
☐ PRO-SCREEN
☐ WINDOWS ☐ GAMES
☐ SUPERTEXT ☐ C VIDEO

FREE SHIPPING – VISA/MC/COD/CHECK

Name .....

Address .....

..... Phone .....

VISA or MC# ..... Exp. Date .....

ZORTECH Inc. 361 Massachusetts Ave, Arlington, MA 02174

Orders & Enquiries Tel: (617) 646 6703

ORDER HOTLINE (800) 848-8408

Number 182 on the Reader Service Card

ZORTECH
BOSTON · LONDON · FRANKFURT · GENEVA

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Program in the fast lane with Borland's new Turbo Pascal 4.0!

Our new Turbo Pascal® 4.0 is so fast, it's almost reckless. How fast? Better than 27,000 lines of code per minute.* That's more than twice as fast as Turbo Pascal 3.0.

4.0 Technical Highlights:

- Compiles 27,000 lines per minute
- Includes automatic project Make
- Supports > 64K programs
- Uses units for separate compilation
- Integrated development environment
- Interactive error detection/ location
- Includes a command line version of the compiler
- Highly compatible with 3.0

For the IBM PS/2™ and the IBM® and Compaq® families of personal computers and all 100% compatibles

4.0 breaks the code barrier

No more swapping code in and out to beat the 64K code barrier. Designed for large programs, Turbo Pascal 4.0 lets you use all 640K of memory in your computer.

4.0 uses logical units for separate compilation

Pascal 4.0 lets you break up the code gang into "units," or "chunks." These logical modules can be worked with swiftly and separately. 4.0 also includes an automatic project Make.

4.0's cursor automatically lands on any trouble spot

4.0's interactive error detection and location means that the cursor automatically lands where the error is. While you're compiling or running a program, you get an error message *and* the cursor flags the error's location for you.

Sieve (25 iterations)

	Turbo Pascal 4.0	Turbo Pascal 3.0
Size of Executable File	2224 bytes	11682 bytes
Execution speed	9.3 seconds	9.7 seconds

Sieve of Eratosthenes, run on an 8MHz IBM AT

Since the source file above is too small to indicate a difference in compilation speed we compiled our CHESS program from Turbo Gameworks to give you a true sense of how much faster 4.0 really is!

Compilation of CHESS.PAS (5469 lines)

	Turbo Pascal 4.0	Turbo Pascal 3.0
Compilation speed	12.1 seconds	35.5 seconds
Lines per minute	27,119	9,243

CHESS.PAS compiled on an 8 MHz IBM AT

Only $99.95

*60-Day Money-back Guarantee***

For the dealer nearest you, or to order now,
Call (800) 543-7543

*Run on an 8MHz IBM AT.

**If within 60 days of purchase this product does not perform in accordance with our claims, call our customer service department, and we will arrange a refund.

All Borland products are trademarks or registered trademarks of Borland International, Inc. Copyright ©1987 Borland International, Inc. BI 1166A

BORLAND

YES! I want to upgrade to Turbo Pascal 4.0 and the 4.0 Toolboxes

If you are a registered Turbo Pascal user and have not been notified of Version 4.0 by mail, please call us at (800) 543-7543. To upgrade if you have not registered your product, just send the original registration form from your manual and payment with this completed coupon to:

Turbo Pascal 4.0 Upgrade Dept., Borland International
4585 Scotts Valley Drive, Scotts Valley, CA 95066

Name _____

Ship Address _____

City _____ State _____

Zip _____ Telephone ( ) _____

This offer is limited to one upgrade per valid registered product. It is good until June 30, 1988. Not good with any other offer from Borland. Outside U.S. make payments by bank draft payable in U.S. dollars drawn on a U.S. bank. CODs and purchase orders will not be accepted by Borland.

For the IBM PS/2™ and the IBM® and Compaq® families of personal computers and all 100% compatibles

†To qualify for the upgrade price you must give the serial number of the equivalent product you are upgrading.

Please check box(es)	Suggested Retail	Upgrade Price†	Serial No.
☐ Turbo Pascal 4.0 Compiler	$ 99.95	$ 39.95	_____
☐ Turbo Pascal Tutor	69.95	19.95	_____
☐ Turbo Pascal Database Toolbox	99.95	29.95	_____
☐ Turbo Pascal Graphix Toolbox	99.95	29.95	_____
☐ Turbo Pascal Editor Toolbox	99.95	29.95	_____
☐ Turbo Pascal Numerical Methods Toolbox	99.95	29.95	_____
☐ Turbo Pascal Gameworks	99.95	29.95	_____

Total product amount $ _____
CA and MA residents add sales tax $ _____
In US please add $5 shipping and handling for each product
Outside US please add $10 shipping & handling for each product $ _____
Total amount enclosed $ _____

Please specify diskette size ☐ 5¼" ☐ 3½"

Payment: ☐ VISA ☐ MC ☐ Check ☐ Bank Draft

Credit card expiration date: _____/_____

Card # _____

PJ 3/4 88

Number 3 on theReader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED